UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

ESTUDIO Y PROTOTIPADO DE LA EFICIENCIA ENERGÉTICA DE UN PANEL SOLAR AUTO-AJUSTABLE LOW-COST

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR: Antonio Sancho Barberán

TUTOR: Joan Josep Fons Cors

2015 - 2016

Resumen

Con este trabajo se pretende abordar la construcción de un mecanismo por el cual nos permita ajustar de forma automática la inclinación de un panel solar fotovoltaico respecto al sol haciendo uso de materiales low-cost para aprovechar la máxima capacidad de generar electricidad del panel.

Además se implementará un sistema software de monitorización por el cual seremos capaces de obtener y almacenar la información generada por dicho panel para su visualización, análisis y ajuste posterior. También se propondrán al final del documento, algunas mejoras al sistema implementado de manera que este sea más productivo.

Palabras clave: Arduino, Placas Solares Fotovoltaicas, Eficiencia Energética

Abstract

With this work I want to build a mechanism that it will allow to adjust us automatically the inclination of a photovoltaic solar panel respect from sun using materials low cost so we will be able to get the maximum capacity to generate electricity from panel.

Furthermore it will implement a monitoring software system so we will be able to get and store information generated for the solar panel for your visualization, analysis and adjustment in the future. Also at the end of the document will be append several improvements to system implemented.

Keywords: Arduino, Photovoltaic Solar Panels, Energy efficiency

Tabla de contenidos

1. Introducción

El desarrollo de este trabajo pretende solucionar un problema existente actualmente en cualquier instalación de paneles solares fotovoltaicos en las cuales dichos paneles se encuentren instalados de manera fija. Este tipo de instalación puede llegar a tener una gran pérdida de eficiencia energética lo largo del día debido a que el sol no siempre está en la misma localización. Con esta esta solución se pretende que cualquier instalación pequeña que tenga instalados paneles solares fotovoltaicos fijos sea capaz de incrementar su eficiencia energética obteniendo finalmente mejores prestaciones en su inversión. Para demostrar lo anteriormente expuesto, se va a desarrollar un pequeño prototipo el cual nos permitirá hacernos una idea sobre que se quiere obtener con este trabajo y de qué manera seria implementado haciendo uso de materiales de bajo coste. También debido a las nuevas tendencias tecnológicas, se va a proponer que dicho prototipo acabe derivando en un dispositivo de IoT debido a que en un futuro casi todos los sistemas van a ser de esta manera. Este dispositivo nos permitirá cubrir dos aspectos importantes en este ámbito. Por un lado será capaz de actuar en tiempo real y por otro lado permitirá historificar la información para un futuro tratamiento.

1.1 Objetivos

Con el presente trabajo se pretende abordar la construcción de dos escenarios diferentes de manera que como se ha mencionado anteriormente este abarcará en ideas de IoT. El primero consistirá en el prototipado de un dispositivo hardware de bajo coste por el cual se proporcione una variación a la inclinación de un panel solar hasta obtener su máxima eficiencia energética fotovoltaica respecto a un panel solar ubicado de manera fija además de ofrecer información del entorno con la cual se creara un histórico.

Por otra parte, el segundo escenario se presentará como una plataforma software la cual hará la función de captación, almacenamiento y análisis de la información recolectada por el dispositivo hardware prototipado anteriormente.

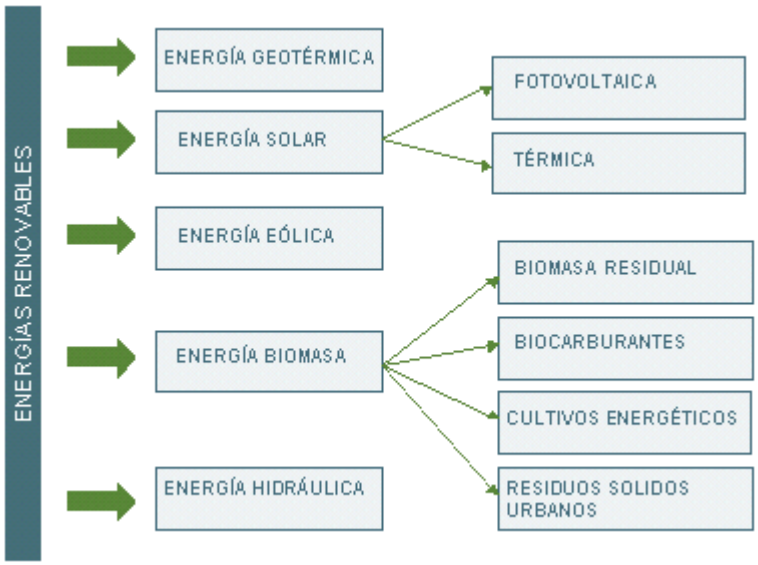
Con esto se pretende diseñar un sistema el cual nos permita obtener la máxima eficiencia energética ya sea a través de este prototipado o bien escalando esta solución a una instalación eléctrica, y podamos decidir en base a los datos capturados y analizados de qué manera podemos intervenir para lograr dicha eficiencia energética.

1.2 Problemática

1.2.1 Energías Renovables

Energía renovable se denomina a aquellas que se obtienen de fuentes naturales inagotables, ya sea por la inmensa cantidad de energía que contienen, o porque son capaces de regenerarse por medios naturales.

Las energías renovables se pueden dividir en diferentes categorías en función de los recursos utilizados para la generación de energía:



Las energías renovables han experimentado un fuerte crecimiento, destacando la energía fotovoltaica y eólica las cuales han experimentado un mayor crecimiento en los últimos años, esto es debido a que se ha incrementado el deseo de sustituir las energías no renovables y contaminantes por las energía renovables a causa del agravamiento del efecto invernadero y el consecuente calentamiento global, acompañado por una mayor toma de conciencia a nivel internacional con respecto a dicho problema para preservar el medio ambiente lo cual ha provocado un movimiento de concienciación en la sociedad para el aprovechamiento de estos recursos naturales como fuente de energía. Se han convertido en la alternativa de nuestro futuro ya que puede liberarnos de la dependencia actual a las de energías contaminantes como el petróleo, carbón, energía nuclear… y su impacto medioambiental es mínimo frente a las energías actuales y antes mencionadas.

Asimismo, economías nacionales que no poseen o agotaron sus fuentes de energía tradicionales (como el petróleo o el gas) y necesitan adquirir esos recursos de otras economías, buscan evitar dicha dependencia energética, así como el negativo en su balanza comercial que esa adquisición representa.

El objetivo que se pretende alcanzar con el uso de energías renovables es que las fuentes contaminantes como petróleo, gas, carbón o la energía nuclear, pasen relegadas a un segundo plano o dejen de ser viables para su uso. Cada vez más somos conscientes de que estas fuentes son recursos limitados y no distribuidos equitativamente por el planeta, y si se continúa utilizándolas a este ritmo las generaciones futuras verán el agotamiento de esas fuentes, comprometiendo el desarrollo de la humanidad.

El consumo de energía es uno de los grandes medidores del progreso y bienestar de una sociedad. El concepto de "crisis energética" aparece cuando las fuentes de energía de las que se abastece la sociedad se agotan. Un modelo económico como el actual, cuyo funcionamiento depende de un continuo crecimiento, exige también una demanda igualmente creciente de energía. Puesto que las fuentes de energía fósil y nuclear son finitas, es inevitable que en un determinado momento la demanda no pueda ser abastecida y todo el sistema colapse, salvo que se descubran y desarrollen otros nuevos métodos para obtener energía: éstas serían las energías renovables.

Un inconveniente evidente de las energías renovables es su impacto visual en el ambiente local. Algunas personas odian la estética de los generadores eólicos y mencionan la conservación de la naturaleza cuando hablan de las grandes instalaciones solares eléctricas fuera de las ciudades.

Otros intentan utilizar estas tecnologías de una manera eficaz y satisfactoria estéticamente: los paneles solares fijos pueden duplicar las barreras anti-ruido a lo largo de las autopistas, hay techos disponibles y podrían incluso ser sustituidos completamente por captadores solares, células fotovoltaicas amorfas que pueden emplearse para teñir las ventanas y producir energía, etc.

1.2.2 Energía Solar

Este concepto se puede definir como el aprovechamiento de la radiación electromagnética producida por el sol para la obtención de energía. La Tierra recibe 174 petavatios de radiación solar entrante (insolación) desde la capa más alta de la atmósfera.

La potencia de la radiación varía según el momento del día, las condiciones atmosféricas que la amortiguan y la latitud. En condiciones de radiación aceptables, la potencia equivale aproximadamente a 1000 W/m² en la superficie terrestre.

Se estima que la energía total que absorben la atmósfera, los océanos y los continentes puede ser de 3.850.000 exajulios por año. En 2002, esta energía en una hora equivalía al consumo global mundial de energía durante un año. La fotosíntesis captura aproximadamente 3000 EJ por año en biomasa, lo que representa solo el 0,08 % de la energía recibida por la Tierra. La cantidad de energía solar recibida anual es tan vasta que equivale aproximadamente al doble de toda la energía producida jamás por otras fuentes de energía no renovable como son el petróleo, el carbón, el uranio y el gas natural.

Este aprovechamiento se puede clasificar en tres grandes grupos:

- *Energía solar fotovoltaica*: consiste en la transformación de la luz solar en energía eléctrica. Este tipo de energía supone, al igual que el resto de energías renovables, un importante beneficio tanto económico como medioambiental. La transformación de la luz solar se realiza por medio de un dispositivo electrónico llamado célula fotovoltaica.
- *Energía solar térmica*: consiste en el aprovechamiento de la radiación solar para la producción de calor. Este calor producido se utiliza para calentar un fluido, normalmente agua, y aprovecharlo para producir agua caliente, calefacción o cualquier aplicación que suponga el calentamiento de un fluido.
- *Energía solar termoeléctrica*: consiste en la utilización de una tecnología que permite utilizar la radiación del Sol para calentar un fluido y producir vapor que se hace pasar por una turbina con el objetivo de generar energía eléctrica.

Una importante ventaja de la energía solar es que permite la generación de energía en el mismo lugar de consumo mediante la integración arquitectónica en edificios. Así, podemos dar lugar a sistemas de generación distribuida en los que se eliminen casi por completo las pérdidas relacionadas con el transporte que en la actualidad suponen aproximadamente el 40 % del total y la dependencia energética.

1.2.3 Energía Solar fotovoltaica

La energía solar fotovoltaica consiste en la obtención de electricidad obtenida directamente a partir de la radiación solar mediante un dispositivo semiconductor denominado célula fotovoltaica

Este tipo de energía aunque era conocida desde la década de 1860, realmente se empezó a utilizar desde su popularización a finales de los años 70 agravado por la crisis del petróleo la cual provoco un cambio importante en la política energética y se prestó atención en dicha tecnología.

Este tipo de energía se empleó tradicionalmente para alimentar aparatos autónomos, abastecer refugios o casas aisladas de la red eléctrica, pero debido a la creciente concienciación de la sociedad sobre lo importante que es para nuestro futuro invertir en energías renovables limpias y seguras, de forma creciente durante los últimos años, se ha potenciado la producción de este tipo de electricidad a gran escala a través de redes de distribución, bien mediante inyección a la red o para autoconsumo doméstico.

Según informes de Greenpeace, la energía fotovoltaica podrá suministrar electricidad a dos tercios de la población mundial en 2030. Y según un estudio publicado en 2007 por el Consejo Mundial de Energía, para el año 2100 el 70 % de la energía consumida será de origen solar.

1.2.4 Eficiencia enérgica

Debido a la actual situación económica que se sigue viviendo en el mundo, tanto gobiernos como empresas como particulares están optando por tomar medidas en aspectos relacionados al consumo y ahorro energético. Es por ello que están buscando soluciones las cuales les ofrezca información con la que puedan tomar decisiones más adecuadas respecto a cómo deben gestionar sus instalaciones eléctricas y equipamiento o de qué manera pueden optimizar los procesos productivos y el empleo de la energía utilizando lo mismo o menos para producir más bienes y servicios.

No se trata de ahorra luz si no de ofrecer mejores servicios consumiendo la menor cantidad de energía. Esto es importante para la industria el cual es uno de los sectores de la sociedad más necesitados del ahorro de energía, ya que su logro supone una mayor competitividad.

Esta eficiencia también puede llegar a ser importante para gobiernos y particulares debido a que un buen diseño de un edificio por ejemplo puede proporcionar aspectos importantes de ahorro de energía y monetario.

1.2.5 Análisis y objetivos

Como se ha mencionado anteriormente la optimización tanto de la producción como del consumo de energía es un factor muy importante para cualquier negocio actual. Esto se debe a que una mala utilización de estos recusros puede hacer que no se aproveche de forma correcta la inversión económica en dichos recursos. Hoy en día y dado la situación geográfica de España es imprescindible que seamos capaces de captar la mayor radiación solar producida por el sol de manera que seamos capaces de transformar a energía eléctrica todo esa radiación que de otra forma se desperdiciaría, la cual hoy en día es un bien muy apreciado y valorado en el mercado.

Pero el mayor problema reside en las instalaciones de usuarios particulares en donde normalmente dichas instalaciones suelen presentarse de manera que los paneles solares

fotovoltaicos son ubicados de manera estática, en los cuales se tiene un ajuste inicial para colocarlo en un ángulo específico respecto al sol, para que así este sea capaz de captar la máxima radiación posible. Por el contra, este tipo de instalación presenta un inconveniente en la cual solo obtendrá su máxima eficiencia cuando el sol incida de manera directa al panel solar fotovoltaico, esta máxima eficiencia solo será posible obtenerla cuando el panel solar y el sol estén alineados de forma perpendicular como puede verse en la figura 1.1, ocurriendo este situación en unas horas determinadas al día. Dado que el sol va cambiando de ángulo a lo largo del día se producirá una situación como la que se puede apreciar en la figura 1.2 en al que vemos como el sol no incide de manera directa sobre el panel solar fotovoltaico penalizando así su eficiencia energética.

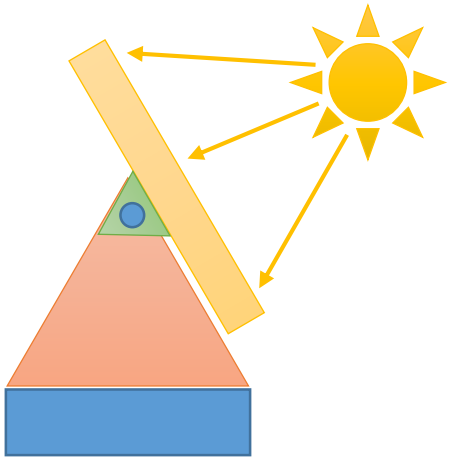


Fig. 1.1 Radiación directa

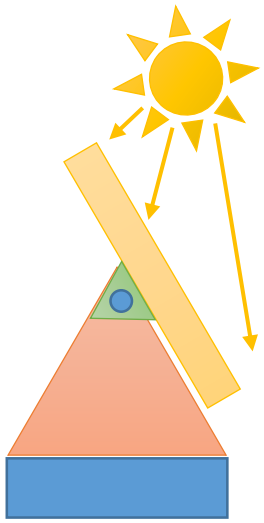


Fig. 1.2 Radiación no directa

Este inconveniente puede ser solucionado si se le añade al panel solar fotovoltaico un mecanismo por el cual sea capaz de seguir la trayectoria del sol consiguiendo de esta manera, que el panel se encuentre un mayor tiempo orientado al sol de forma perpendicular obteniendo una mayor eficiencia y productividad, dado que de este modo no se limitara tanto el número de horas en la cual el panel está obteniendo radiación solar directa. Como podemos se puede apreciar en las siguientes figuras, al añadir este mecanismo al panel solar fotovoltaico se consigue que la radiación del sol incida de manera directa durante un periodo de tiempo más largo, consiguiendo que se incremente la eficiencia obtenida.

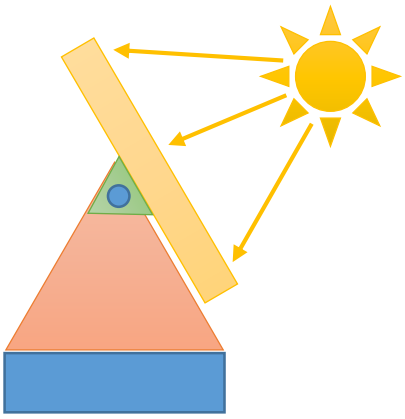


Fig. 1.3 Radiación directa

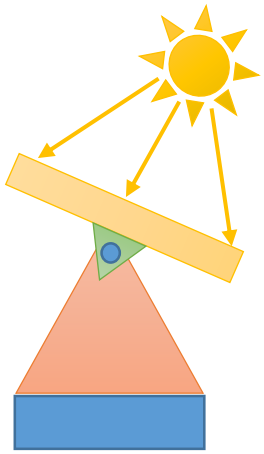


Fig.1.4 Radiación directa

1.3 Situación Actual

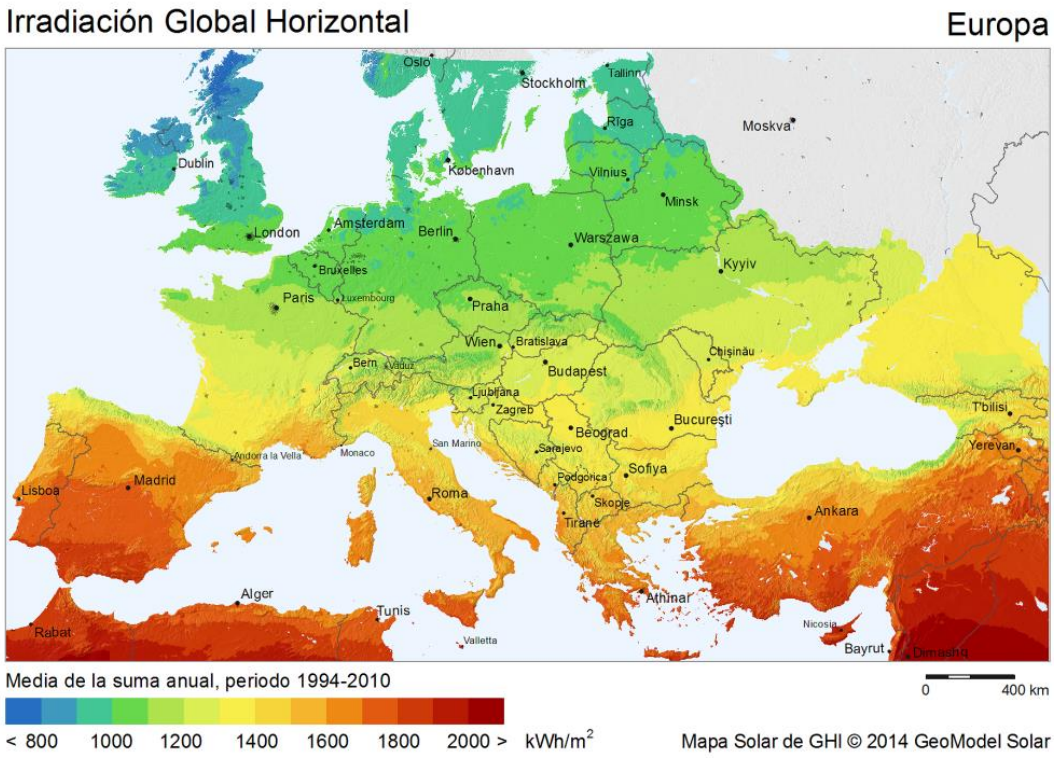
En la actualidad la energía solar fotovoltaica se encuentra en un proceso de expansión debido a la búsqueda de nuevas formas de producir energía limpia y segura. Para aprovechar la energía radiada por el sol, debemos hacer uso de dispositivos llamados paneles solares los cuales están compuestos de células fotoeléctricas que permiten transformar la energía lumínica (fotones) en energía eléctrica (flujo de electrones libres) mediante el efecto fotoeléctrico.

Debido al principio de funcionamiento de los paneles solares fotovoltaicos, estos deben de recibir la máxima radiación solar posible de forma directa en sus células, teniendo que hacer estudios y cálculos del lugar donde vaya a ser ubicado para evitar posibles sombras o reflejos de objetos cercanos y penalizando su eficiencia a la hora de producir energía.

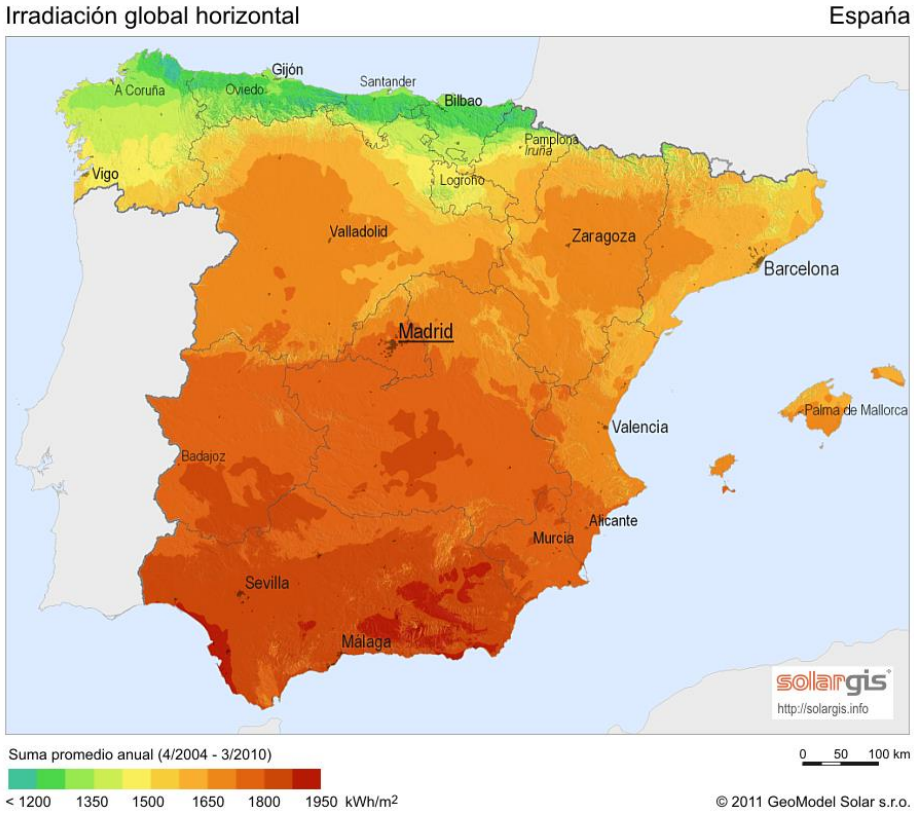
A la hora de hacer una instalación de paneles solares fotovoltaicos, la mayoría de veces estos son instalados de tipo estático/fijo, esto quiere decir que el panel una vez haya sido ubicado en su localización definitiva este habrá sido instalado cumpliendo ciertos parámetros con el objetivo de obtener su máxima eficiencia energética, y así de esta manera consiguiendo unos costes de instalación y mantenimiento muy bajos para el propietario.

Esta situación descrita tiene una importante desventaja, la cual consiste en la limitación a la hora de capturar la radiación solar por parte del panel debido a que el sol no siempre se encuentra en la misma posición a lo largo del día, esto se traduce en que habrá únicamente una franja horaria al día en la cual el panel tendrá su máxima eficiencia energética, y el resto del día no obtendrá una eficiencia óptima. También hay situaciones en las cuales el panel puede llegar a perder eficiencia tales como días nublados o nubosos, las noches e incluso las diferentes estaciones del año pueden llegar a afectar su eficiencia debido al comportamiento que tiene el planeta respecto al sol.

En España tenemos una principal ventaja respecto a otros muchos países, debido a la ubicación geográfica en el que nos encontramos, esta ventaja se traduce en que recibimos más radiación solar al año que otros países y por tanto podríamos tener la capacidad de aprovechar toda esa radiación recibida para producir energía más limpia y segura que las actuales fuentes de energía disponibles.



Todo ello contribuyó a que España fuera inicialmente uno de los primeros países a nivel mundial en investigación, desarrollo y aprovechamiento de la energía solar. Gracias a una legislación favorable, España fue en 2008 uno de los países con más potencia fotovoltaica instalada del mundo, con 2708 MW instalados en un solo año. La potencia instalada de energía solar fotovoltaica en España alcanzaba los 4672 MW a finales de 2014. Sin embargo, regulaciones legislativas posteriores frenaron la implantación de esta tecnología.



1.4 Finalidad

Como consecuencia de todo lo descrito anteriormente, con este estudio se pretende abordar una solución de bajo coste la cual permitirá obtener una mejora en la eficiencia energética de los paneles solares respecto a los paneles instalados de manera estáticos. Esta solución buscara de forma autónoma obtener la máxima luminosidad procedente del sol a lo largo del día suponiendo un incremento en la productividad energética.

Con la solución propuesta es este estudio se quiere demostrar que sería viable para cualquier empresa o particular adaptar un panel solar que tenga instalado de manera estático en su propiedad a un panel el cual sea capaz de hacer un seguimiento del sol a lo largo del día de forma automático obteniendo así un incremento en la productividad enérgica de su instalación, todo ello realizado con componentes y dispositivos de bajo coste.

Para la monitorización de la solución al seguidor solar se añadirá una arquitectura software la cual permitirá un conexión de todos los paneles solares a un host para recolectar y almacenar la información que estos proporcionen y posteriormente poder realizar estadísticas a partir de los datos recolectados, de esta manera pudiendo hacer observaciones sobre como de eficiente llega a ser el panel a la larga.

2. Contexto Tecnológico

Desde este punto se pretende describir las tecnologías que van a ser necesarias para el desarrollo del trabajo. Debido a que el futuro avanza en dirección a la interconexión de los dispositivos y sistemas, en este trabajo se ha optado por querer experimentar en mayor o menor medida en el concepto de IoT (Internet of things). Este concepto se refiere a la interconexión digital de objetos cotidianos con internet. Alternativamente, IoT es el punto en donde en el futuro se conectaran a internet más "cosas u objetos" que personas. De esta manera todos estos dispositivos serán capaces de ofrecer información de su entorno generando así otro concepto conocido como Big Data. Este otro concepto hace referencia al almacenamiento de grandes cantidades de datos y a los procedimientos usados para encontrar patrones repetitivos dentro de esos datos

2.1 Tecnología Hardware

Para este prototipo ha sido necesario hacer uso de una variedad de componentes sobre los cuales parte la solución. En principio partíamos de dos componentes principales que eran el panel solar fotovoltaico y el regulador de tensión, estos son la base a partir del cual se implementa realmente la solución planteada.

El panel solar fotovoltaico empleado tiene como características principales una potencia nominal de 5 W y una tensión de 18 voltios el cual es válido para proporcionar energía a baterías de 12 voltios.

Debido a que el panel proporciona una diferencia de potencial mayor que el que las baterías requieren, es necesario añadir un dispositivo, este es llamado regulador de tensión y es el encargado de adaptar la tensión proporcionada por el panel, a una tensión estable y apta de salida para transferirla a la batería.

A partir de los elementos básicos descritos anteriormente podemos empezar a plantear la construcción del sistema de control y monitorización del prototipo. Para ello se necesitaran dos tipos de dispositivos diferentes.

Por un lado nos harán falta un conjunto de sensores y actuadores, cuáles serán los elementos que capturen las medidas físicas obtenidas del entorno y las transformen en señales eléctricas o bien actúen sobre el entorno físico de alguna manera, los diferentes tipos de sensores que emplearemos para este prototipo serán de temperatura y humedad, de voltaje, de intensidad y de luminosidad. Cada uno de ellos nos proporcionara información diferente del entrono la cual nos ayudará a tomar futuras decisiones.

Por otro lado será necesario un dispositivo que sea capaz de obtener la información proporcionada por los sensores y tenga la capacidad de procesar dicha información con el objetivo de tomar las decisiones oportunas. Para esta tarea el mejor dispositivo que se ajusta es un micro-controlador o un mini-pc, dado que nos permite hacer las tareas anteriormente descritas dado que estos tipos de dispositivos tienen puertos de entrada/salida, y también tienen una CPU de mayor o menor potencia la cual depende del dispositivo, pero nos permite hacer cálculos y operaciones para tratar los datos obtenidos. Este tipo de dispositivo tiene una característica muy importante para este tipo de propósito dado que para su funcionamiento requiere de un consumo reducido. Entre las opciones más destacadas de este tipo de dispositivo para este propósito podemos encontrarnos la plataforma de hardware libre Arduino o bien el mini-ordenador Raspberry Pi principalmente. Existen otras muchas alternativas para este tipo de dispositivo, pero he decidido centrarme en estas dos opciones debido a que estas dos son las más extendidas y a su vez disponen de una mayor comunidad y documentación en Internet,

aclarar que estas dos placas tienes dos filosofías muy distintas y por eso es necesario estudiar bien el escenario y ver cuál de las dos se adapta mejor a él.

2.1.1 Sensores

Para este prototipo serán necesarios unos dispositivos que nos permitan obtener las magnitudes físicas o químicas del entorno y las transforme en señales eléctricas, este tipo de dispositivo se le conoce como Sensor.

Estas magnitudes medibles pueden ser temperatura, presencia, viento, campos magnéticos, flujos de líquidos, calidad del aire, propiedades de los líquidos, vibraciones, posiciones geográficas, altitud, presión y un largo etc. Estos dispositivos toman la información en bruto por lo que se requieren de otros que permitan capturar y actuar en referencia a los datos obtenidos.

2.1.2 Actuadores

Un actuador es un dispositivo capaz de ejecutar y/o recibir una orden del controlador y realizar una acción sobre un aparato o sistema concreto (encendido/apagado, subida/bajada, apertura/cierre, etc)

Para hacer posible el movimiento del panel, será necesario hacer uso de un dispositivo el cual nos permita modificar dicha inclinación respecto al sol. Para esta tarea tenemos diferentes alternativas, como pueden ser servomotores o motores.

Los servomotores son dispositivos semejantes a un motor con la principal diferencia que incorpora un circuito de control con el cual podemos especificarle una posición concreta dentro de su ángulo de giro, para este tipo de dispositivos podemos encontrar de dos tipos: rotan 180 grados o rotan 360 grados. Por otra parte tenemos los motores convencionales de corriente continua o corriente alterna, para hacer funcionar este tipo de motor será necesario añadir un dispositivo controlador adicional el cual nos permitirá controlar de manera sencilla su dirección y velocidad de giro.

2.1.3 Controlador

Es una computadora utilizada en la ingeniería automática o automatización industrial, para automatizar procesos electromecánicos según la programación recibida durante su montaje o instalación y la información que recibe, tales como el control de la maquinaria de la fábrica en líneas de montaje, atracciones mecánicas, etc.

A diferencia de las computadoras de propósito general, un controlador está diseñado para múltiples señales de entrada y de salida, rangos de temperatura ampliados, inmunidad al ruido eléctrico y resistencia a la vibración y al impacto. Los programas para el control de funcionamiento de la máquina se suelen almacenar en memorias no volátiles. Un controlador es un ejemplo de un sistema de tiempo real «duro», donde los resultados de salida deben ser producidos en respuesta a las condiciones de entrada dentro de un tiempo limitado, de lo contrario no producirá el resultado deseado.

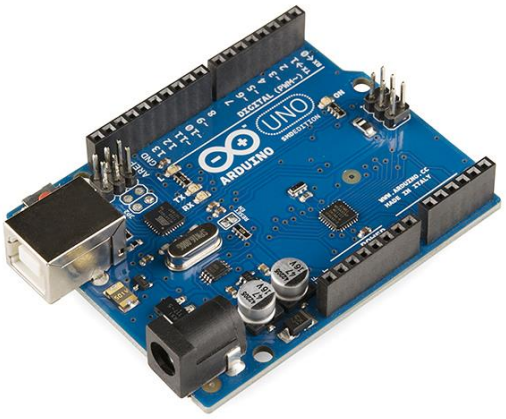
Para este trabajo las alternativas de controladores existentes que mejor se ajustan son Arduino y Raspberry Pi, aunque cada una de ellas están orientadas para distintos objetivos y situaciones.

Plataforma libre Arduino

La plataforma libre Arduino es un dispositivo hardware de propósito general el cual está basado en un micro-controlador Atmel AVR, dispone de puertos de entrada/salida por los cuales podemos tomar información del entorno, también tiene un entorno propio de desarrollo el cual nos permite llevar a cabo proyectos multidisciplinares de una manera rápida, sencilla y con unos costes muy bajos.

Debido a la gran popularidad que ha ganado en los últimos años para hacer multitud de pequeños proyectos, existe una gran comunidad en internet en donde sus usuarios hacen por mejorar el ecosistema de esta placa de desarrollo hardware como por ejemplo creando nuevas librerías para nuevos dispositivos o nuevas funcionalidades, creando placas de expansión, etc.

También debido a su gran aceptación para la enseñanza o aprendizaje en la electrónica y programación, existen fabricantes como por ejemplo BQ que han decido hacer su propia versión de esta placa dotándola de algunas mejoras, pero conservando su esencia principal e incitar de esta manera la enseñanza de estas tecnologías a los más pequeños.



La plataforma Arduino proporciona las siguientes ventajas respecto a otros sistemas ya ensamblados con micro-controlador:

- **Asequible**: Las placas Arduino son más asequibles comparadas con otras plataformas de micro-controladores.
- **Multi-Plataforma**: El software de desarrollo de Arduino tiene soporte en los sistemas operativos Windows, Macintosh OSX y Linux. La mayoría de los entornos de desarrollo para micro-controladores están limitados a Windows.
- **Entorno de programación simple y directo**: El entorno de programación de Arduino es fácil de usar para principiantes y lo suficientemente flexible para los usuarios avanzados.
- **Software ampliable y de código abierto**: El software Arduino está publicado bajo una licencia libre y preparado para ser ampliado por programadores experimentados, no se pagan licencias de ningún tipo.
- **Hardware ampliable y de Código abierto:** Arduino está basado en los micro-controladores ATMEGA168, ATMEGA328 y ATMEGA1280. Los planos de los módulos están publicados bajo licencia Creative Commons, por lo que diseñadores de circuitos con experiencia pueden hacer su propia versión del módulo, ampliándolo u optimizándolo.

Raspberry Pi

El mini-ordenador Raspberry Pi, es un ordenador reducido de muy bajo coste desarrollado por la fundación Raspberry Pi con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas, pero debido a sus enormes capacidades es posible emplear este pequeño ordenador en muchos otros campos de aplicación que no requieran de unas altas prestaciones.

Esta placa está compuesta por una CPU la cual varía dependiendo del modelo que se escoja, así también de una memoria RAM y un procesador gráfico. Para el almacenamiento dispone de una ranura para tarjetas micro SD en la cual se alojará el sistema operativo y los datos de usuario.

Para su funcionamiento es necesario hacer la instalación de un sistema operativo con el que podamos gestionar el hardware además de ofrecernos un alto nivel de desarrollo gracias a que podemos emplear cualquier lenguaje de programación, el Sistema Operativo más conocido para esta placa y al cual la fundación da soporte es Raspbian, esta distribución viene derivada de otra distribución muy conocida como es Linux Debian, también en los últimos modelos de placa la fundación ha dado soporte a Windows 10 en una versión reducida pensada para el desarrollo.

Como principales ventajas de esta plataforma se podrían destacar:

- **Hardware similar a un ordenador**: Al tener un Sistema Operativo totalmente funcional al estar basado en Debian, tiene la capacidad de ejecutar la mayoría de programas que hay disponible para este sistema.
- **Mayor potencia que un micro-controlador**: Al estar basado su hardware en un ordenador, dispone de características superiores a las que puede tener un micro-controlador.
- **Menor consumo respecto a un ordenador normal**: Debido a los componentes por los cuales está construida, esta placa consume únicamente 5 w respecto a los 30 w que puede llegar a consumir un ordenador portátil.

2.1.5 Conectividad

Para la interacción de los dispositivos hardware encargados de recoger información del entorno con los servicios software que estarán alojados en otra máquina se pueden presentar varias alternativas de comunicación donde se deberá elegir de forma correcta el tipo y características de tecnología que necesitamos emplear para la tarea de intercambiar información entre dos máquinas remotas, esto se conoce como M2M (Machine to Machine).

La comunicación puede ser de dos naturalezas diferentes: a través de cable (RS232, PLC, Ethernet, RTC, RDSI, ADSL, etc) o bien a través de redes inalámbricas (GSM/UMTS/HSDPA, Wifi, Bluetooth, RFID, Zigbee, UWB, etc).

Principalmente seguiré dos estrategias para realizar la interconexión de los elementos involucrados en el trabajo. Esto permitirá recibir información desde una gran cantidad de dispositivos gracias a la fácil escalabilidad que puede llegar a tener el sistema.

Por un lado tendremos la estrategia de **Dispositivo + Pasarela** en donde la comunicación se puede desarrollar mediante un protocolo conocido y bastante extendido como puede ser RS232, RS485 o radio (868 MHz) donde existirá un dispositivo el cual tendrá comunicación directa a una maquina pasarela la cual recibirá la información generada por el dispositivo.

Por otro lado tendremos una **Red de sensores + Pasarela** donde la comunicación de los dispositivos se realizara a través de una red de comunicación local por medio de WIFI o Ethernet por la cual interconectara a todos ellos y finalmente comunicara al exterior a través de la pasarela.

Adicionalmente existe una solución la cual se conoce como **Standalone**, en donde los dispositivos tendrán una comunicación autónoma y esta se realizara por medio de Módems 3G/GRPS. Esta alternativa se emplea cuando el acceso a una red de datos es compleja debido al entorno que rodea el emplazamiento del dispositivo, es por ello que se recurren a redes móviles para la transferencia de información.

En los últimos años, debido al aumento en la tendencia del IoT (Internet of Things), muchos de los protocolos de comunicación tradicionales están siendo mejorados con el objetivo de obtener mejores distancias de transmisión y menores consumos energéticos. El aspecto energético ha sido durante muchos años un elemento de batalla en las comunicaciones precisamente por los altos consumos de estos componentes, y los diseñadores y fabricantes actualmente lo tienen en el punto de mira para continuar su mejora.

2.1.6 Regulador de carga

Un regulador de carga es un dispositivo encargado de controlar constantemente el estado de carga de las baterías así como de regular la intensidad de carga con el fin de alargar la vida útil de las baterías. Controla la entrada de corriente que proveniente del panel solar y evita que se produzcan sobrecargas y sobredescargas profundas en la batería que puede reducir el rendimiento de la batería o la esperanza de vida, y puede suponer riesgo de seguridad.

Los paneles solares pequeños de 1 a 5 Watios no requieren la instalación de ningún regulador dada su baja potencia, pero muchos paneles solares entregan entre 16 a 20 voltios, esto podría llegar a estropear la batería por un efecto de sobrecarga, ya que esta necesita unos 14,5 voltios para una carga adecuada

Los fabricantes de paneles los hacen con voltajes superiores a los 12 voltios como las baterías debido a que salvo en condiciones ideales, los paneles no producen su tensión máxima, de modo que estos son diseñados para proporcionar más tensión ya que al trabajar en condiciones no ideales entregarán la mayor parte del tiempo tensiones cercanas a los 12 voltios, y en el caso de que existiera mucho sol, entonces ya se encargaría de trabajar el regulador de carga.

2.2 Tecnología Software

Una vez se tiene la plataforma hardware sobre la cual se va a trabajar, es importante elegir correctamente que lenguajes software serán necesarios para implementar toda la lógica del proceso, esta elección es una de las más importantes y complicadas, dado que una mala elección puede llegar a hacernos perder mucho tiempo y recursos, sea por, bajos conocimientos en el lenguaje elegido al inicio y se requiera de una curva de aprendizaje, puedan surgir complejidades a la hora de desarrollar la solución o se creen desarrollos que terminan siendo complejos y de difícil mantenimiento.

2.2.1 Software dispositivo hardware de seguimiento (FIRMWARE ARDUINO)

Para la implementación del software encargado de hacer el seguimiento solar, se hará uso del propio entorno de desarrollo de Arduino. Este es gratuito y multi-plataforma y es posible obtenerlo desde la propia página web del proyecto. Este entorno hace uso de un lenguaje propio basado en el lenguaje de programación de alto nivel Processing que es similar a C++. Al estar basado en C, soporta todas las funciones del estándar C y algunas de C++.

También se hará uso de las librerías necesarias para utilizar el sensor de temperatura y el servomotor.

2.2.2 Software de interconexión, persistencia y consulta.

Para la implementación de este paso se utilizara el lenguaje de programación Java por dos motivos fundamentales, el primero de ellos es porque es multiplataforma, esto quiere decir que tendremos la capacidad de ejecutar el software generado en cualquier sistema operativo siempre y cuando dispongamos de una máquina virtual Java. Y el segundo es más por motivo personal, por el hecho de desarrollar en un lenguaje con el que estoy más familiarizado, así de esta manera pudiendo llegar a ofrecer una mejor y eficaz solución.

En algún momento se barajó la idea de desarrollar el software bajo otras tecnologías como pueden ser Node.JS o Python pero esta idea fue finalmente descartada por la falta de conocimiento en dichas tecnologías, por lo que el inicio se habría retrasado y es posible que no se hubiera podido llegar a la solución planteada, debido en parte en que también afecto en esta decisión la falta de tiempo para desarrollar el trabajo, dado que el aprendizaje de un nuevo lenguaje requiere de una curva en la que se tienen que invertir muchas horas dependiendo del lenguaje, por estos motivos decidí usar una tecnología conocida, en la que me siento cómodo, y tengo el conocimiento suficiente como para llegar a presentar una solución, por último, como opinión personal, opino que dichos lenguajes no me parecen una buena solución, debido a que son lenguajes de una moda actual, en donde son empleados en multitud de proyecto, pero no pueden llegar a ofrecer una coherencia y orden en el software desarrollado a no ser que se haga uso de un Framework.

Retomando el hilo sobre la tecnología que será empleada en este trabajo, su implementación se hará haciendo uso de diferentes tecnologías, en donde también haremos uso del entorno de desarrollo de software Eclipse la cual nos ayudara a poner un orden al código generado, nos validara el código generado y nos ayudara a generar los paquetes de despliegue. Para la creación de la API de comunicaciones con el hardware y la API REST se utilizó el lenguaje Java en su versión 8.

Gestión del proyecto y librerías

Como en todo proyecto a veces es necesario utilizar software ya implementado por otros usuarios ahorrando tiempo en desarrollar cosas por las que otros ya han pasado antes, es aquí en donde para la gestión de las librerías necesarias en el trabajo se optó por usar Maven. Maven es una herramienta de software que sirve para gestionar y construir proyectos el cual nos permite obtener librerías de un repositorio o también generar los paquetes de salida para el despliegue. Debido a lo complejo que resulto obtener todas las librerías necesarias al inicio, de esta manera, esta herramienta software me facilitó el obtener todas las librerías y dependencias que estas requerían a su vez, y me permitió generar los ficheros de despliegue para hacer las pruebas con todos los recursos necesarios para su ejecución.

Framework para persistencia de datos

Debido a la naturaleza del trabajo es necesario guardar información en una base de datos, para este cometido existen muchos métodos software por el cual solucionarlo, para este caso, en un principio pensé utilizar para la capa de persistencia de los datos la herramienta software ORM Hibernate la cual permite hacer relaciones entre los atributos de las tablas a objetos mediante el uso de anotaciones, pero debido a su elevada curva de aprendizaje y del tiempo que requiere, decidí buscar alternativas la cuales tuvieran características similares pero fueran más fáciles y rápidas de implementar. En esta búsqueda encontré una herramienta de persistencia de datos llamada MyBatis la cual tiene características similares que Hibernate a diferencia de que MyBatis no hace un mapeo de objetos java a tablas sino que ofrece el acceso a métodos a través de sentencias SQL, resulta más sencillo de implementar, pudiéndose llegar a controlar mejor las sentencias SQL ejecutadas, no como ocurre en Hibernate que como suele decirse hace magia a la hora de generar dichas sentencias SQL.

API Rest

Para la recepción y consulta de información se ha implementado una API REST, esto se define como una arquitectura software por la cual se implementa un servicio sin estado en el que se ofrecen recursos y no servicios como tal y sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes, por la cual podemos gestionar de una forma sencilla el intercambio de información entre sistemas heterogéneos. Al ser un servicio sin estado quiere decir que perdemos los datos cada vez que enviamos una nueva petición, además con los recursos ofrecidos en el servicio se podrán realizar diferentes acciones por medio de las cabeceras del propio protocolo HTTP como son GET, POST, PUT, DELETE. Para ello el servicio elegido para implementar el servicio ha sido Jersey debido en parte a la gran documentación existente al respecto y la simplicidad a la hora de utilizarlo.

Jersey es un Framework para el desarrollo de web services RESTful el cual proporciona soporte para JAX-RS. Proporciona una API propia que extiende las herramientas de JAX-RS con características y utilidades adicionales para promover el desarrollo de servicios y clientes RESTful simplificados, además permite que los desarrolladores puedan extender Jersey a sus necesidades.

Intercambio de información

Para el intercambio de información entre las distintas partes software existen dos diferentes alternativas, JSON o XML. Cada uno de ellos presenta diferentes características en la estructuración de su contenido.

Por un lado esta JSON el cual tiene un formato de texto ligero basado en un subconjunto del lenguaje de programación JavaScript y puede ser leído por cualquier lenguaje de programación. Para la creación de los mensajes JSON tenemos dos estructuras básicas:

- Una colección de pares de nombre/valor. (objeto, estructura, tabla hash).
- Una lista ordenada de valores. (Arrays, vectores, listas).

Debido a que estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra.

Por otro lado tenemos a XML, se puede definir como un lenguaje de marcas utilizado para almacenar datos de forma legible. Permite definir una gramática para estructurar documentos grandes. Se ha adoptado como un estándar para el intercambio de información estructurada entre diferentes plataformas.

Este tipo de mensaje ha sido criticado por su nivel de detalle y complejidad. El mapeo del árbol básico de XML por medio de los lenguajes de programación puede ser difícil debido a que resulta más complejo preparar la estructura mediante la cual se accederá al contenido, consumiendo más recursos que otros formatos.

Por la complejidad de los mensajes XML a la hora de procesarlo y tienen mayor tamaño debido a la forma de estructura la información, se optó por emplear el formato de mensaje JSON debido a que es más ligero su procesamiento y es capaz de procesarse en una gran cantidad de lenguajes. Es fácil generar mensajes en este tipo, y su envió no requiere de grandes transferencias de información.

API Hardware

La comunicación de los dispositivos hardware a la maquina pasarela se puede realizar por medio de dos tecnologías diferentes implementadas con tecnología Java.

Por un lado será posible conectar el hardware directamente a la maquina a través de un cable USB, esta conexión hará uso del protocolo de comunicaciones RS232 el cual nos proporciona una conexión serie directa entre el dispositivo hardware y la maquina pasarela. Este tipo de tecnología tiene un claro inconveniente, es necesario un cable físico de conexión entre los dispositivos y únicamente es posible conectar un dispositivo a un puerto USB físico, o bien mediante la ayuda de HUB llegar hasta la máxima capacidad de dispositivos que puede gestionar el bus USB.

Para solventar la problemática que puedan surgir a la hora de conectar dispositivos por medio del protocolo serie, también se ha optado por implementar la opción de un servicio en red, este servicio está a la escucha en un puerto en la red en la cual está conectada la maquina pasarela y de esta forma, los nodos que estén conectados a dicha red ya sea por una conexión Ethernet o Inalámbrica serán capaces de comunicarse con la máquina y transmitirle la información recogida.

2.2.3 Visualización y representación de los datos

En la actualidad, con los avances tecnológicos que han surgido, se cuenta con bases de datos que pueden almacenar una gran cantidad de información, esta información a su vez necesita ser procesada para poder extraer conocimiento de ella, de esta necesidad de análisis surge la tecnología de Minería de Datos o Data Mining (DM).

Sin embargo, no podemos hablar de DM sin antes tratar el tema de inteligencia de Negocios o Business Intelligence (BI). La Inteligencia de Negocios se define como el proceso de analizar los bienes y datos acumulados con el objetivo de extraer conocimientos de ellos, es una disciplina que ayuda al proceso de toma de decisiones y aquí es donde BI resulta ser la solución más adecuada cuando contamos con una gran cantidad de información y a través de ella se pueden generar escenarios, pronósticos y reportes que funcionan como auxiliar para buscar las soluciones más acertadas, traduciéndose finalmente en ventajas competitivas.

En la actualidad existen muchas soluciones orientadas a BI, y estas a su vez pueden ser utilizadas en diferentes áreas. Contar con una de estas soluciones es muy beneficioso y permite obtener de forma casi instantánea de información importante sobre la situación del negocio.

DM es un componente esencial de BI, abarca una serie de técnicas, algoritmos y métodos cuyo fin es la explotación de grandes volúmenes de datos con vistas al descubrimiento de información previamente desconocida y que puede servir de ayuda en el proceso de toma de decisiones, de ahí la estrecha relación que existe entre estas dos disciplinas.

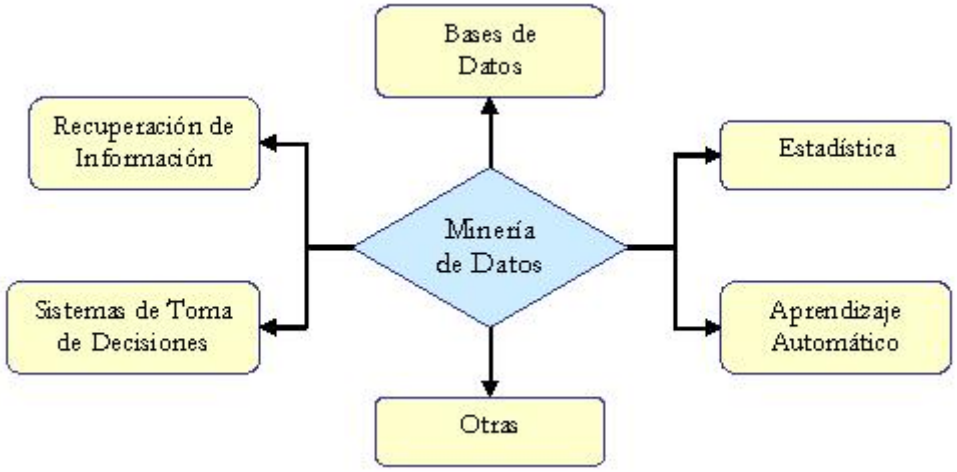
DM también se define como el proceso de descubrir patrones en los datos. Otra definición sería DM es el proceso eficiente, no trivial, de extraer información valiosa (patrones y tendencias) de una gran colección de datos.

Las principales razones que impulsaron el crecimiento de DM se describen a continuación:

- Las nuevas necesidades de análisis de grandes volúmenes de datos.
- Gran parte de la información es histórica.
- El notable crecimiento de la web en estos últimos tiempos.
- La competitividad entre las organizaciones.
- El surgimiento de una gran variedad de software y herramientas para llevar a cabo DM.

- La necesidad de predecir la información futura.
- La toma de decisiones se puede basar en datos históricos.

DM se nutre de la siguientes disciplinas: aprendizaje automático, base de datos, estadística, gestión de organizaciones, teoría de la decisión, visualización.



Como parte fundamental en este trabajo y como se ha mencionado anteriormente, gracias a los datos almacenados es posible identificar las debilidades o fortalezas y de esta manera tomar acciones oportunas que nos beneficie de una u otra manera.

Es por ello que para la representación de la información almacenada en este trabajo no se realizara con la ayuda de ninguna herramienta software ya existente para estas tareas debido a que ellas están pensadas para volúmenes de información más elevada. Por estos motivos hacer uso de una herramienta de este tipo para este trabajo podría llegar a resultar ineficiente, porque realmente a través de este prototipo se quiere mostrar lo que seriamos capaces de llegar a conseguir y no hacer un estudio comercial sobre lo tratado en el trabajo. De esta manera lo que se pensó fue implementar una pequeña página web en la cual se visualicen los datos que han sido almacenados a partir de los dispositivos hardware y lleguemos a ser capaces de ver una breve introducción a la idea que se quiere exponer con el título del trabajo.

Para esto se implementara un portal web basado en Tecnología AngularJS el cual nos proporcionara por pestañas una presentación de la información almacenada ya sea a través de visualización de gráficas, tablas, descarga de datos en fichero, etc. De esta manera seria posible el crear herramientas y técnicas que nos permitan monitorizar obteniendo una visión de forma más rápida de los datos.

Angular es un Framework de desarrollo basado en Javascript que es usado de manera frecuenta para la creación de aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.

Este Framework adapta y amplía el HTML tradicional para servir mejor contenido dinámico a través de un data binding bidireccional que permite la sincronización automática de modelos y vistas. Como resultado, AngularJS pone menos énfasis en la manipulación del DOM y mejora la testeabilidad y el rendimiento.

Angular sigue el patrón MVC de ingeniería de software y alienta la articulación flexible entre la presentación, datos y componentes lógicos. Con el uso de la inyección de dependencias, Angular lleva servicios tradicionales del lado del servidor, tales como controladores dependientes de la

vista, a las aplicaciones web del lado del cliente. En consecuencia, gran parte de la carga en el backend se reduce, lo que conlleva a aplicaciones web mucho más ligeras.

2.2.4 Base de datos

Por sus características, una base de datos es uno de los elementos primordiales en todo sistema de información; a diferencia de otros medios de almacenamiento, como los archivos, las bases de datos poseen estructuras y componentes internos que organizan los datos de manera eficaz, el acceso a estos datos es permitido a través de un lenguaje de consultas que puede ser de dos tipos: SQL (Structure Query Languaje), o NoSQL (Not only Structure Query Languaje). Las bases de datos son administradas por un motor de base de datos o un sistema de administración de bases de datos y los diferentes SGBD (Sistemas Gestores de Base de Datos) serían los responsables de almacenar y gestionar toda esta información de madera adecuada.

Hoy en día, la importancia que tiene el uso de base de datos es cada vez más incuestionable, las mismas son utilizadas en todo tipo de organizaciones: tanto con propósitos académicos, como en entidades comerciales, y almacenan toda clase de información. Los SGBD deben afrontar ciertos desafíos para gestionar de manera eficiente este volumen masivo de información y extraer conocimiento de estos repositorios, existen arquitecturas de diferentes tipos, diseñadas con características específicas para facilitar las tareas de almacenamiento y gestión de los datos.

Arquitecturas

Podemos encontrar diferentes arquitecturas de SGBD:

- **Centralizada**

  Una base de datos centralizada es aquella que almacena grandes cantidades de datos en un solo lugar físico o nodo central. En una arquitectura Cliente-Servidor, los datos son accedidos por los usuarios a través de terminales que solo muestran resultados.

  Los SGBD centralizados, tienen las siguientes desventajas:

  - Si el sistema de base de datos falla, se perdería la disponibilidad y procesamiento de la información que posea el sistema.
  - Difícil sincronización para su recuperación.
  - No es posible distribuir la carga de trabajo entre varios nodos.

- **Distribuida**

  Este tipo de base de datos tiene una particularidad que la diferencia de las demás, y es que los datos están almacenados en distintas máquinas interconectadas entre sí, formando parte del sistema.

  Cada uno de las maquinas que integran dicho sistema reciben el nombre de localidad o nodo, por lo que toda la información se encuentra distribuida en diferentes espacios lógicos, a diferencia de una base de datos centralizada cuya característica principal es que la información se almacena en un solo nodo; hay que destacar que a pesar de que los datos se encuentran distribuidos, el conjunto de nodos que forman parte del sistema representan una única base de datos.

- **Federada**

La arquitectura federa fue introducida por primera vez por Denis Heimberge en el año 1982, está basada en el intercambio coordinado de información organizada por modelos que describen conceptos y comportamientos comunes. Es una colección de diversos sistemas de base de datos cooperativos y autónomos en la que existe una interfaz común, que es utilizada por los usuarios para acceder al sistema. Las Bases de Datos Federadas (BDF) están formadas por varios esquemas unificados; cada uno describe una porción de la base de datos que aparenta ser una sola, sin embargo, es una colección de sistemas independientes, heterogéneos y autónomos. Una BDF aparenta ser una BD normal y corriente, pero no tiene existencia física, es una vista lógica. Debido a su naturaleza distribuida y heterogénea, esta arquitectura es adecuada para ser utilizada en escenarios con problemas de complejidad causados por ambientes heterogéneos, como en IoT, donde hay una enorme cantidad de dispositivos u objetos, que ofrecen servicios muy diversos y localizados en diferentes lugares.

Hay que destacar que una base de datos federada, es diferente de una distribuida, porque a pesar de que ambas arquitecturas comparten ciertas semejanzas, (como por ejemplo el hecho de que los datos está repartidos en diferentes ubicaciones, y que los nodos están interconectados a un nodo central), sin embargo su diseño lógico es diferente, un SGBD distribuido es de tipo Top-down, es decir, existe una sola base de datos distribuida en diferente localidades y que se rigen por las mismas reglas, por lo tanto, carecen de independencia, además los usuarios la perciben como una única base de datos, aunque la misma esté distribuida; mientras que un SGBD federado es de tipo Botton-up, es decir un conjunto de nodos autónomos e independientes interconectados a un nodo central con estructura.

Un modelo de datos es el responsable tanto de determinar el tipo de estructura lógica que será utilizada en una base de datos, como de establecer el modo en el que los datos serán organizados, almacenados y estructurados.

Modelos de datos

Para la definición de dicha estructura el modelo de datos utiliza lenguajes para hacer consultas que pueden ser de dos tipos:

- **SQL**

  SQL es un lenguaje para hacer consulta utilizado como estándar predeterminado para los Sistemas de Gestión de Bases de Datos Relacionales (SGBDR), cuya función principal es la de gestionar de manera adecuada las operaciones de actualización, creación de esquemas y modificaciones en el sistema relacional.

  SQL al ser un estándar relacional, muchos de sus conceptos pertenecen al modelo relacional.

- **NoSQL**

  Debido al notable crecimiento que han tenido los sistemas informáticos en las últimas décadas atribuido al desarrollo de nuevas aplicaciones y recursos web, el volumen de datos asociados a estas aplicaciones también ha crecido.

  En escenarios como estos, en donde se manejaba una gran cantidad de información, los SGBD relacionales se mostraban poco escalables e ineficientes; para afrontar estos problemas del modelo relacional fueron buscadas soluciones alternativas, una de ellas fue la de aumentar el número de servidores, sin embargo, esta solución no resultaba tan sencilla de aplicar por lo estructurado que es un SGBDR.

Las bases de datos NoSQL surgieron como alternativa para solucionar los problemas del modelo relacional; hay que destacar que el propósito de NoSQL no es el de sustituir el modelo relacional, más bien, surge como alternativa para escenarios en donde se requería una arquitectura mucho más flexible y menos estricta que la del modelo relacional.

La función principal de los sistemas NoSQL es la gestión y el almacenamiento de datos semiestructurados, cada uno de estos procesos se hace de forma independiente, por lo que los datos no necesitan tener un modelo predefinido ni encajar en tablas relacionales, lo que hace que el sistema sea más flexible en escenarios con ambientes heterogéneos.

Un dato semiestructurado es un tipo de datos que no poseen definición de tipos, ni conceptos de variables o atributos, y no están organizados mediante un patrón determinado.

Ventajas de NoSQL frente a SQL

- **Evitar complejidad innecesaria:** las base de datos relacionales proporcionan una serie de funcionalidades relacionadas con la consistencia de datos (el llamado ACID) que las base de datos NoSQL no implementan, lo que hace que el rendimiento sea mayor al reducir el número de comprobaciones que tiene que hacer el sistema.
- **Alta Escalabilidad:** añadir y eliminar servidores a una base de datos NoSQL es más fácil porque no se requiere ninguna configuración adicional en el resto de servidores.
- **Evitar el mapeo a objetos:** muchas aplicaciones trabajan con una estructura de datos muy simple (únicamente una lista de pares clave/valor), por lo que no se benefician del modelo relacional que proporcionan las bases de datos tradicionales.
- **Sacrificar fiabilidad frente a rendimiento:** en algunas situaciones puede ser interesante sacrificar la fiabilidad para ganar rendimiento.
- **Bases de datos + capa de caché contra sistemas creados desde cero pensando en alta escalabilidad:** cuando empezaron a crecer los requerimientos de las base de datos se creó lo que se llama una capa de cache, que permite aumentar la capacidad de carga de las base de datos. El problema es que este sistema es un parche que está llegando ya a su límite de capacidad, por eso se plantea utilizar un sistema que se haya creado desde cero y que desde su planteamiento sea diseñado para proporcionar un alto rendimiento/escalabilidad.

Solución propuesta para base de datos

El criterio para decidir que SGBD seleccionar, es saber cuál será la finalidad que se le va a dar, ya que por ejemplo, para IoT lo que se pretende es priorizar las inserciones. Una de las posibles soluciones sería trabajar con un sistema distribuido de base de datos de forma que cada nodo del sistema sería capaz de recibir una parte del volumen de información que se genera

Las razones por las que un ambiente centralizado no se ajusta a IoT se mencionan a continuación:

En primer lugar, si se toma en cuenta que la información proviene de distintas fuentes, realizar minería de datos en un sistema centralizado y sobre datos distribuidos sería muy complejo.

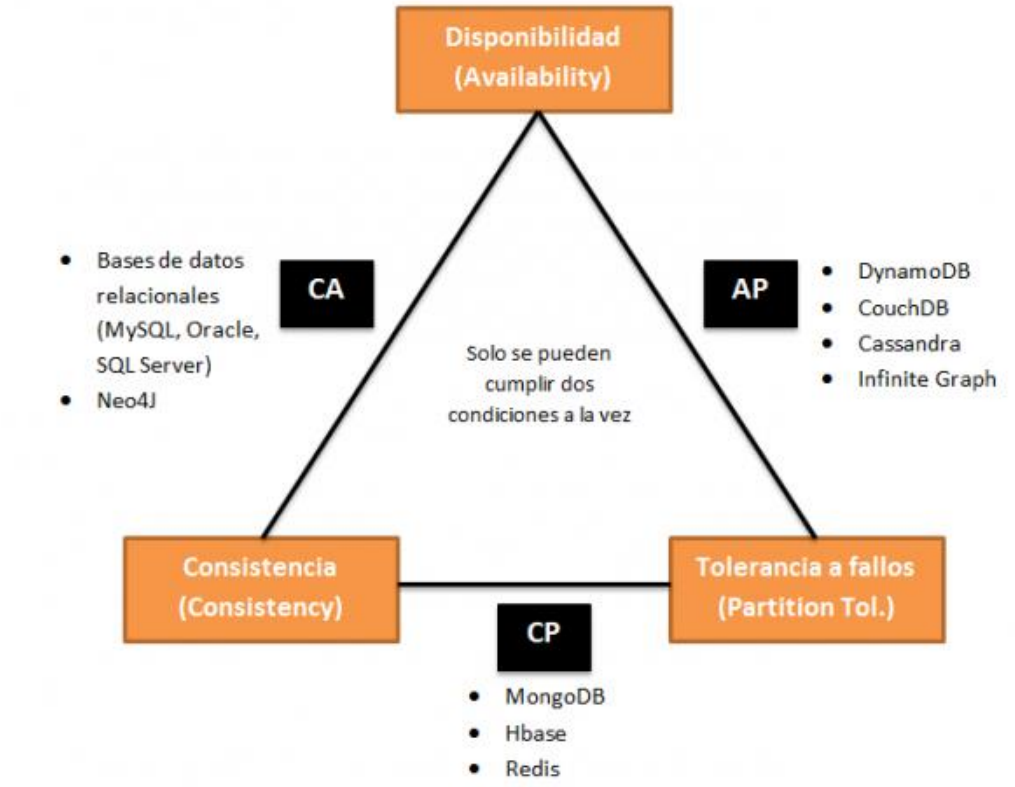
En segundo lugar, la cantidad de información que genera IoT es muy elevada y requiere ser procesada en tiempo real, por lo que haría falta un nodo central con una capacidad de cálculo demasiado grande.

En tercer lugar, por temas de seguridad y privacidad no es recomendable almacenar toda la información en un único lugar.

Y por último, enviar toda la información de los sensores provocaría sobrecarga en la red.

Por todo esto una arquitectura distribuida no solo evitaría todos esos problemas, sino que también reduciría la complejidad global del sistema.

Uno de los problemas que tienen los SGBDR se derivan de aplicar el teorema CAP; en el siguiente gráfico se muestran varios SGBD y como estos se ven afectados por ese teorema:



El primer grupo (CA) sería el de aquellas bases de datos que no tienen tolerancia a particionado, dentro de este grupo están incluidas los SGBDR, ya que no están preparados para tener nodos en redes distintas. El segundo grupo (CP) es el de aquellas que han sacrificado una disponibilidad alta, y el último grupo (AP), son aquellas que no implementan ACID y por tanto han perdido la consistencia de datos.

Para IoT, de las tres características de Teorema CAP la más importante es la de disponibilidad, ya que en IoT los datos son masivos y en tiempo real, por lo que es más crítico que el sistema permita la escritura y lectura de datos con rapidez, por eso se puede descartar el grupo CP, ya que no tiene esta característica fundamental.

De las dos características restantes, la más importante sería la Tolerante a Particiones, en virtud de que esta permite que la red esté configurada en varios nodos que pertenezcan a redes distintas y cada nodo gestiona los objetos que estén en su zona, por lo que en un escenario IoT esto sería más factible.

Ahora solamente quedaría disponible AP, hay que tener en cuenta, que aunque estos SGBD no implementan ACID, y por tanto, no hay consistencia de datos, no quiere decir que los datos sean siempre inconsistentes, ya que la mayoría de estos sistemas utilizan técnicas alternativas (Como puede ser la actualización en segundo plano, la replicación, etc.) para implementar lo que se

llama consistencia eventual que significa que los datos que no son consistentes "eventualmente" lo serán.

Para el desarrollo de este trabajo será necesario usar una base de datos para almacenar toda la información recibida desde los dispositivos hardware, para esta tarea finalmente se ha elegido la base de datos relacional MySQL aun teniendo en consideración lo descrito arriba y en lo poco beneficioso que puede resultar este tipo de base de datos, pero los principales motivos que me llevaron a hacer uso de este tecnología fueron la falta de tiempo y la falta de conocimientos en otros gestores, anteriormente ya había realizado algunos proyectos con esta tecnología así de esta manera ha sido más rápido y sencillo el llevar a cabo el prototipo. También al estar basada en una base de datos relacional, tiene semejanzas a muchas de las tareas que realizo diariamente en el trabajo con Oracle, por lo que crear estructuras de consulta o inserción no resultaba nada complejo.

2.2.5 Tomcat - Contenedor de aplicación web

Tomcat es un software desarrollado con Java (con lo cual puede funcionar en cualquier sistema operativo, con su máquina virtual java correspondiente) que sirve como servidor web con soporte de servlets y JSPs.

Tomcat no es un servidor de aplicaciones, como JBoss o JOnAS. Este incluye el compilador Jasper, que compila las JSPs convirtiéndolas en servlets. Además el motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache.

Actualmente Tomcat va por su versión 8 estable.

3. Caso de Estudio

A continuación se van a exponer las etapas de análisis y diseño del trabajo planteado en las cuales se presentaran como va a abordarse la implementación y de qué manera tiene que llevarse a cabo. Dado que este trabajo tiene partes involucradas en el concepto de IoT, es importante señalar por qué se hace hincapié en el diseño del dispositivo hardware que será construido.

3.1 Análisis

Debido a que una instalación tradicional de paneles solares fotovoltaicos puede resultar ineficiente por su grado de incidencia directa del sol, se ha querido plantear una optimización a este problema por medio de este trabajo, para ello se indicará como se ha creado una serie de mecanismos que van a permitir obtener la máxima radiación solar de manera directa y de esta manera obteniendo una mayor productividad energética. Esta solución pretende ser aplicable a cualquier instalación de paneles solares fotovoltaicos de cualquier particular o pequeña empresa. Si se quisiera aplicar a una instalación grande debería de realizarse algunas mejoras en el diseño que permitirá operar en dicho escenario de aplicabilidad.

Para conseguir esta mejora el trabajo se basará en dotar a un panel solar fotovoltaico de un mecanismo el cual le permita realizar cierto seguimiento a la posición del sol, así de esta manera se obtendría una mayor productividad energética debido a que será posible capturar la máxima radiación posible traduciéndose en un aumento del aprovechamiento de la energía solar. Gracias a esto seremos capaces de capturar mucha más radiación que antes y durante más horas diarias incrementando las prestaciones de la instalación. Para ello será necesario de un software mediante el cual tengamos la capacidad de observar las medidas obtenidas por los sensores y actuar sobre los dispositivos encargados de conseguir dicha inclinación del panel solar.

Para llevar a cabo el seguimiento solar durante el mayor número de horas de sol disponibles, será necesario equipar al panel solar con una serie de dispositivos los cuales permitan llevar a cabo dicha acción. Esto será un conjunto de dispositivos y procesos los cuales se encargaran de obtener a través de los sensores unos valores del entorno y en consecuencia realizaran la acción que mejor sea dependiendo la programación especificada. Para ello será necesario adaptar o sustituir el soporte fijo del panel solar por otro que le permita a este cierta movilidad y de esta forma conseguir orientar el panel solar fotovoltaico en dirección al sol de manera perpendicular.

Como se ha mencionado anteriormente será necesario de un software mediante el cual sea posible monitorizar toda la información del entorno obtenida por el dispositivo. Así de esta manera seremos capaces de mantener el panel de manera perpendicular al sol durante el mayor tiempo posible y por otra parte, debido a las nuevas tendencias que están surgiendo, se pretende aplicar el concepto de IoT a este trabajo, es por ello será necesario la creación de una API por la cual podamos comunicar el dispositivo hardware a otra máquina remota la cual será la encargada de recibir la información de todos los posibles dispositivos conectados. Esta conexión podrá ser realizada de dos formas diferentes, o bien por una conexión directa por un cable USB o bien por una conexión de red a un servicio. De esta manera se crearía un histórico de información por la cual podríamos llegar a ser capaces de obtener información predictiva.

Esta API permitirá también que cualquier dispositivo que esté conectado a una red local Ethernet o WIFI puedan establecer una conexión a ella, así de esta manera permitiendo el envío de la información obtenida por dichos dispositivos conectados de esta manera y no existiendo una limitación en número de dispositivos por esta forma .

Debido al tipo de trabajo que se quiere presentar será necesario almacenar la información en una base de datos dado que resultaría imposible hacer un estudio con todos los datos recogidos en tiempo real, por este motivo para poder realizar un análisis a la información recogida será necesario almacenarla en una base de datos. Así de esta manera a través del histórico de datos almacenados se pueden llegar a realizar estudios comparativos pudiendo de esta manera obtener resultados sobre el comportamiento obtenido en dicha mejora añadida al panel solar fotovoltaico. Para la tarea del almacenamiento de datos se utilizara una base de datos MySQL, no es la más óptima para estos menesteres pero por limitación temporal y conocimientos sobre ella va a ser la elegida. En ella se almacenará toda la información enviada de forma que posteriormente podrá ser consultada por un portal web o bien por una aplicación externa dado que el sistema de mensajes empleado es en un formato generalizado JSON siendo este un formato soportado por la mayoría de las tecnologías existentes.

Debido a que tiene que existir un punto de interconexión de todo el sistema, será necesario implementar un software cuya función principal será encargarse de recibir todos los mensajes enviados mediante POST desde las diferentes APIs conectadas y a continuación este las persistirá en la base de datos, además también se encargara de recibir peticiones de consulta vía GET y devolverá respuestas con los datos solicitadas. Este tipo de solicitud serán realizadas por el portal web o bien por aplicaciones externas que requieran de datos del sistema. Este software debe de ser accesible por todos debido a que va a ser un servicio y será desplegado en un servidor Tomcat.

Para la representación de los datos obtenidos por la monitorización al panel solar fotovoltaico, se implementara un portal web el cual nos permitirá mediante gráficas y tablas representar los datos obtenidos por los dispositivos de manera que sean entendibles y claros con un simple vistazo, en donde se pudieran ver los valores medios, máximos y mínimos registrados para los diferentes días, meses o años para cada uno de las magnitudes del entorno registradas por los dispositivos.

3.2 Diseño

3.2.1 Soporte ajustable del panel solar

Para esta tarea será necesario crear una superficie que permita rotar sobre un eje central anclado a un soporte en donde estará alojado el dispositivo controlador cuya función principal será encargarse de monitorizar todo el sistema. En la figura siguiente se puede ver un boceto sobre la idea descrita y como tiene que ser implementado.

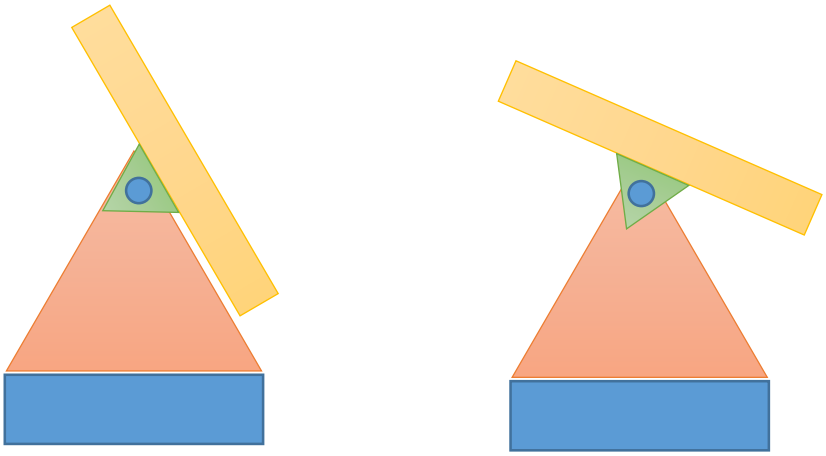


Fig. 3.2.1 Ejemplo de soporte inclinable

Para conseguir el movimiento de dicha superficie superior será necesario de un motor o servomotor el cual nos permita modificar la inclinación respecto la ubicación del sol a lo largo del día de forma que la radiación impacte de forma directa. En la figura inferior se puede apreciar una idea de cómo irá acoplado el servomotor en el soporte principal, permitiendo de esta forma que el panel pueda obtener cierta inclinación respecto al sol.

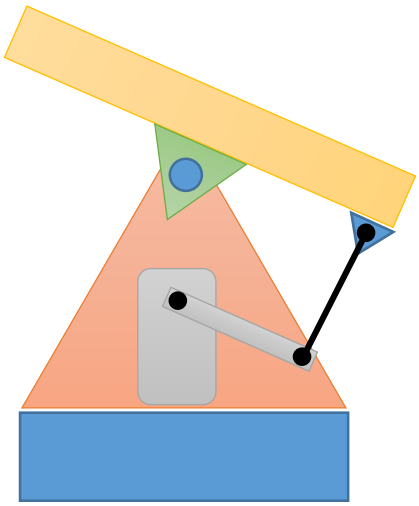


Fig. 3.2.2 Ubicación Servomotor

3.2.1 Dispositivo hardware para monitorización.

Para esta tarea es necesario hacer uso de diversos dispositivos con el objetivo de medir el entorno físico en el que se encuentra el panel. Para esta tarea podemos diferenciar entre los sensores necesarios para medir el entorno y un controlador que se encargue de obtener y procesar los valores obtenidos de los sensores.

Sensores

Por llevar a cabo este trabajo será necesario utilizar dos sensores de luminosidad los cuales nos permitirán obtener la radiación procedente del sol y de esta manera poder ajustar la posición del panel respecto a la luminosidad del sol. Estos sensores deben de ser colocados de forma precisa de modo que siempre reciban la máxima luminosidad y no exista ningún elemento que interfiera en la luminosidad que reciban.

La mejor ubicación para colocar estos sensores corresponde a la parte superior e inferior del panel respectivamente (se puede ver un ejemplo en la figura 3.2.3), de esta manera es más fácil hacer un seguimiento a la posición del sol, dado que al tener la tierra un movimiento natural de rotación, la posición del sol varia y de esta forma la luminosidad directa del sol será diferente para cada uno de los sensores. De este modo el sensor que esté ubicado en una posición en la que la radiación del sol no incida de forma directa obtendrá unos valores más bajos que el sensor que este ubicado de forma más directa a la radiación del sol. Así al producirse esta diferencia, podemos saber que el panel no está recibiendo la máxima radiación solar posible y podemos actuar en consecuencia.

En condiciones de baja luminosidad como los dos sensores recibirán la misma luminosidad, el panel no modificara su inclinación evitando situaciones del día que puedan existir nubes, tormentas, noche, etc.

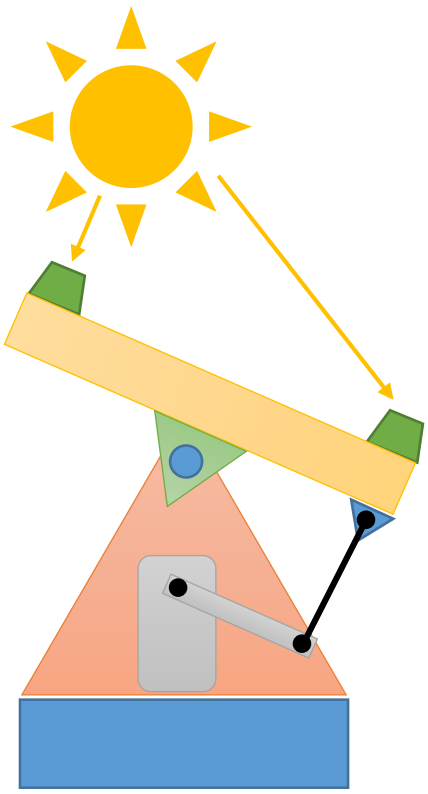


Fig. 3.2.3 Ubicación de sensores luminicos

En la figura de la izquierda, los dos trapecios verdes representan los dos sensores de luminosidad ubicados en el soporte del panel. Esta disposición es la mejor manera debido a lo explicado anteriormente.

Abajo se presenta como es fisicamente una resistencia LDR o fotorresistencia.

Fig. 3.2.4 Representación de una resistencia LDR

Para los sensores de luminosidad se harán uso de resistencias LDR o fotorresistencia (como las que se muestran en la figura 3.2.4) las cuales corresponden a un componente electrónico cuya resistencia disminuye con el aumento de intensidad de luz incidente. El valor de resistencia eléctrica de un LDR es bajo cuando hay luz incidiendo en él (puede descender hasta 50 ohms) y muy alto cuando está a oscuras (varios megohmios).

También sería interesante dotar a esta instalación de un sensor de temperatura y humedad (puede apreciarse en la figura 3.2.6), este sensor sería ubicado en el centro del soporte (puede apreciarse en la figura 3.2.5), debajo del propio panel solar. De forma que podamos medir de forma directa a que temperaturas y humedades está expuesto dicho panel. El principal motivo es que los paneles solares empiezan a disminuir la tensión a medida que se van calentando por el sol, haciendo que dejen prácticamente de funcionar en climas de muy altas temperaturas. Por esta razón puede convenir mantener una monitorización de dichos valores con el objetivo de

enfriarlos como por ejemplo dejando que el aire circule por debajo de ellos por medio de algún ventilador, modificando la inclinación, etc.

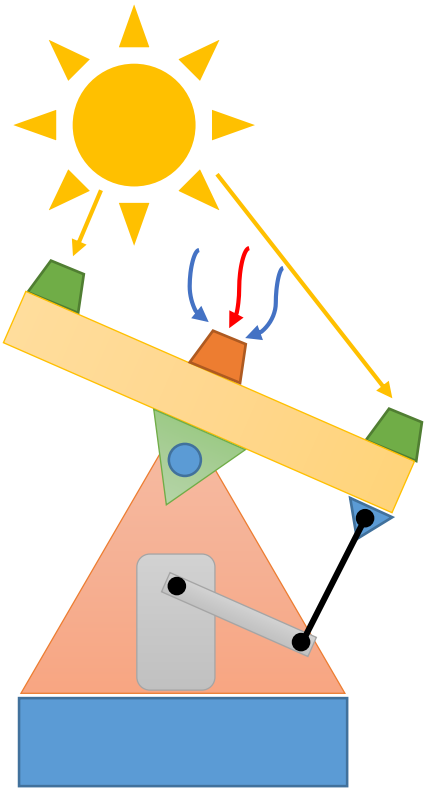


Fig. 3.2.5 Ubicación del sensor de temperatura

Fig. 3.2.6 sensor de temperatura

En la figura superior se aprecia el sensor de temperatura y humedad.

En la figura de la izquierda, el trapecio anaranjado representa la ubicación del sensor respecto al soporte. Al ser ubicado debajo de la placa solar, este obtendrá la misma temperatura y humedad a la que está sometido el panel solar.

Para obtener los valores de producción energética del panel solar, necesitaremos instalar dos tipos de sensores adiciones. Por un lado será necesario dos sensores que nos mida por un lado el voltaje generado por el panel solar y por otro el voltaje que tiene la batería, para esto se ha implementado un divisor resistivo el cual siempre nos devuelve un valor no superior a 5,1 voltios, donde esos 5,1 voltios representarían los 18 voltios que puede llegar a generar el panel solar. En la figura inferior puede verse el esquema correspondiente al divisor resistivo.

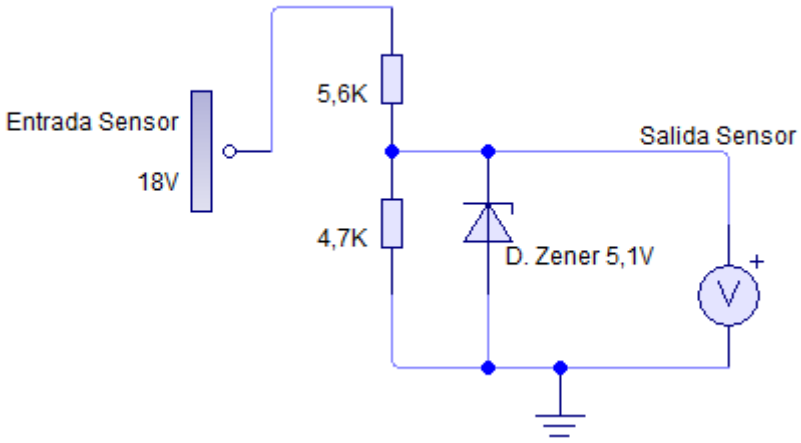


Fig. 3.2.7 Divisor resistivo como sensor de voltaje

Por otro lado se necesitara un sensor para medir la corriente generada por el panel y que llega a la batería. Para esta tarea usaremos una placa que incorpora el sensor de corriente ACS712 de 5ª (En la figura 3.2.8 puede verse como es el sensor comentado). Este tipo de sensor se les conoce como sensores de efecto hall, dado que fue descubierto por el físico Edwin Hall en 1879, es caracterizado por la creación de un campo eléctrico en un material conductor o semiconductor por el cual circula una corriente, este es atravesado perpendicularmente por un campo magnético el cual desvía las cargas y genera una diferencia de potencial y un campo eléctrico en el material. El campo magnético genera la fuerza de Lenz sobre el conductor o semiconductor, causando la desviación de los portadores de carga que se mueven a través del material, dando origen a la diferencia de potencial en los extremos del conductor, el voltaje es directamente proporcional a la intensidad de campo magnético aplicado.

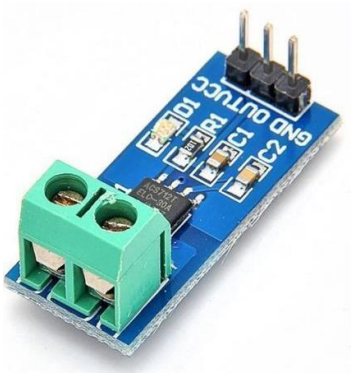


Fig. 3.2.8 Sensor ACS712 de 5A

Controlador

Para la tarea de obtener los datos registrados por los sensores y procesarlos se hará uso de una placa electrónica llamada Arduino Nano (puede verse en la figura 3.2.9) la cual está construida con un micro-controlador ATmega328 y tiene 8 puertos de entrada analógica los cuales son necesarios para el tipo de sensor que estamos empleando. Esta placa se encargara de obtener la información de los diferentes sensores y actuar respecto a la lógica que se le haya programado. Como por ejemplo controlar el servo-motor dependiendo de la luminosidad total obtenida. Al tener un puerto USB es fácil de conectar a un dispositivo pasarela el cual se encargara de recopilar toda la información.

Fig. 3.2.9 Controlador Arduino Nano

Al disponer de 8 pines analógicos, todos los sensores menos el de temperatura y humedad irán conectados a esta placa por dichos puertos, debido a que todos ellos proporcionan sus lecturas en unos rangos de voltajes y como este micro-controlador tiene 10 bits de resolución este nos da un rango de 0 a 1024 valores en el margen de los valores de 0v a 5v. El principal motivo para

usar este tipo de dispositivo es su gran versatilidad para estos escenarios además de tener un consumo mínimo lo cual es muy importante para este tipo de escenario.

El sensor de temperatura y humedad ira conectado a uno de los 14 pines digitales que tiene dado que este tipo de sensor emplea un protocolo de comunicación especial llamado 1-Wire el cual está basado en un bus, un maestro y varios esclavos de una sola línea de datos en la que se alimentan.

En la figura siguiente puede verse un resumen del conexionado de los diferentes sensores a la placa controladora.

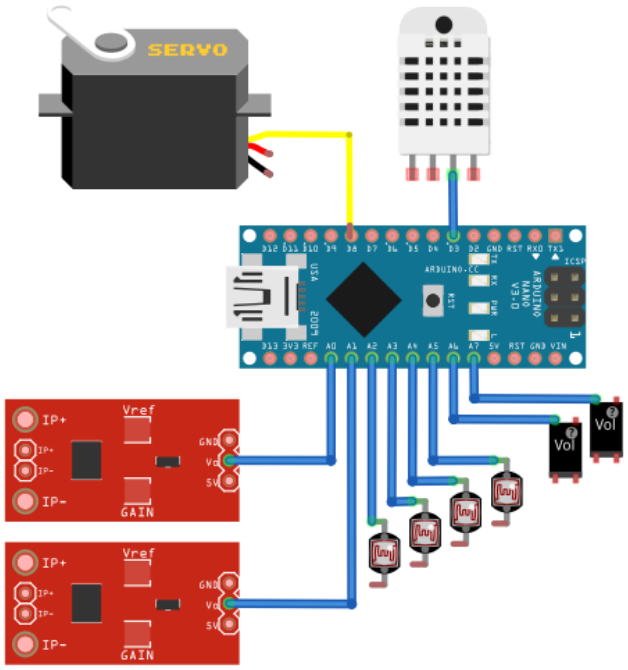


Fig. 3.2.10 Esquema conexionado del sistema de monitorización

Regulador de Carga

Dado que la placa que usamos para el prototipo es de 5w no es necesario añadir un regulador de control, pero si se añadirá un mecanismo para estabilizar la tensión generada por el panel solar dado que este puede llegar a tener picos de 22 voltios. También se le añadirá algún elemento de protección con el cual evitaremos que haga el camino inverso la energía generada y se añadirá algunos de los sensores mencionados anteriormente para poder llevar a cabo la monitorización.

3.2.2 API software – Hardware

Este capa software estará implementada en tecnología java, y será la que nos permita obtener información del entorno desde los diferentes dispositivos que puedan haber conectados.

La arquitectura de esta capa software permitirá que una maquina en la que este desplegado y funcionando esta API permita la conexión de tantos dispositivos hardware como limitación exista por parte del protocolo de comunicación del puerto USB, además también se activara un servicio el cual estará a la escucha de nuevos dispositivos hardware clientes que estén conectados dentro de su red de área local. Así de esta manera se podrá recibir información tanto de dispositivos que estén conectados a la red local de manera física como Ethernet o bien de dispositivos que estén conectados a una red inalámbrica como por ejemplo una red de sensores

conectados vía WIFI. A continuación se puede apreciar por medio de esta figura grafica la idea planteada anteriormente.

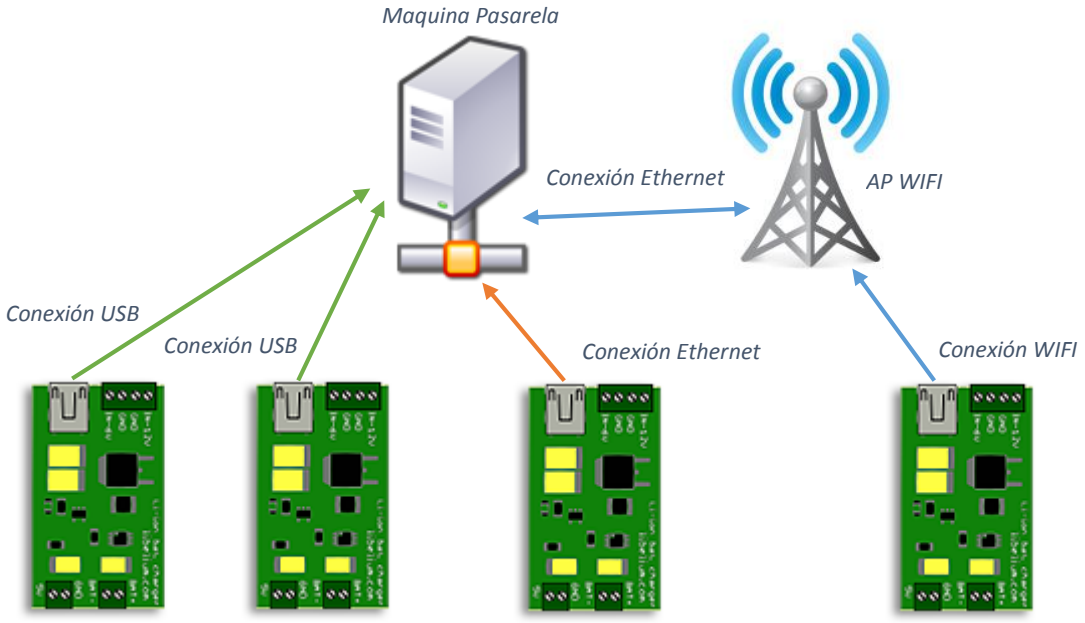


Fig. 3.2.11 Arquitectura dispositivos hardware y API

La máquina pasarela en la cual sea desplegada la API puede ser tanto un ordenador normal como un mini ordenador Raspberry Pi, el único requisito que deben de cumplir es que sean capaces de ejecutar programas en java además de que disponga de conexión a internet. Este último requisito es muy importante dado que la información que este recibe de los dispositivos hardware, debe de ser enviada al servidor central el cual se encargara de persistir dichos datos en la base de datos.

Toda esta comunicación, tanto si esta se realiza por conexión Serie como si se realiza por conexión de red, se realizara mediante mensajes en formato JSON con una estructura propia. Se intentara hacer los mensajes lo más reducido posible con el objetivo de que su tamaño sea menor y de esta manera se conseguirá una mayor velocidad de transferencia entre el dispositivo hardware y la API de comunicación.

3.2.3 Servidor central REST

Para tener un mayor control de toda la arquitectura se creará una estructura jerarquizada donde se desarrollará un servidor central el cual estará formado por un servicio REST que permitirá recibir mensajes a través del protocolo HTTP vía peticiones POST y también será capaz de recibir peticiones GET por las cuales se devolverá la información solicitada.

Este servidor será el encargado de hacer diversas tareas, entre ellas pueden ser la persistencia a la base de datos de toda la información recibida mediante los mensajes JSON enviados por POST de todas las maquinas pasarela, también se encargará de servir las peticiones de consulta recibidas por HTTP con GET ya sea para la representación de los datos en el portal web o también podría ser una aplicación externa que requiera la información para hacer otras tareas. A continuación se puede apreciar por medio de la siguiente figura la idea que se quiere implementar.

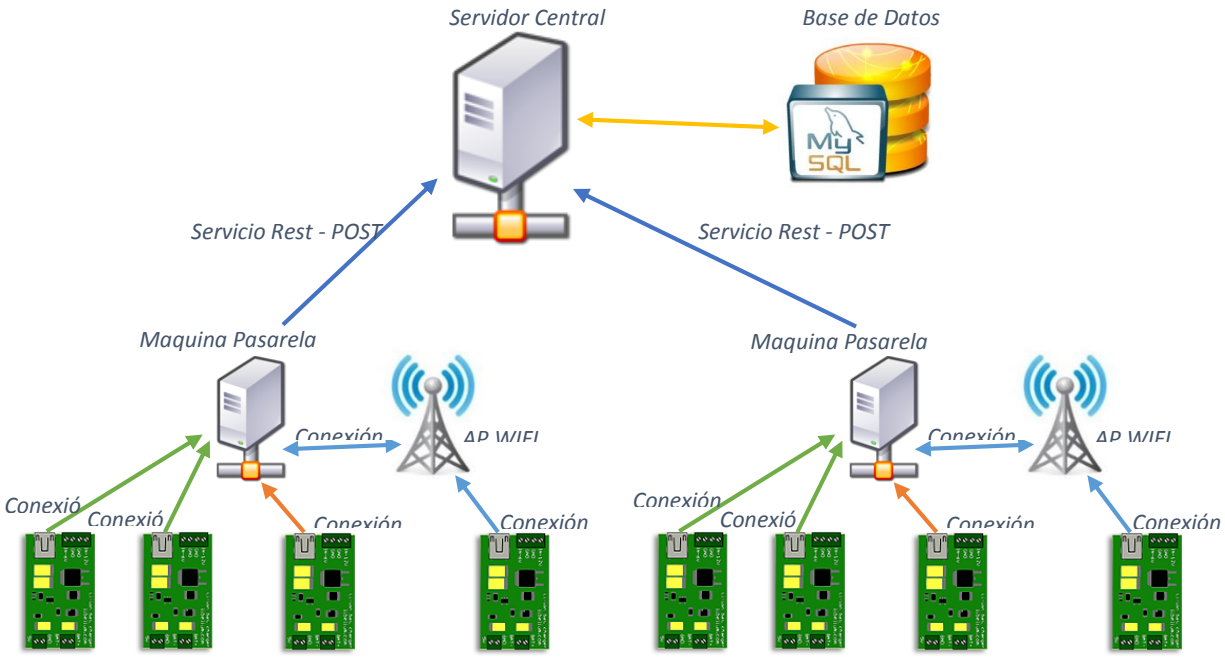


Fig. 3.2.12 Arquitectura dispositivos, API y servidor central

Este servidor central tendrá la capacidad de ofrecer una página web debido a que tiene que ser desplegado en un servidor Tomcat, por este motivo se aprovechara y almacenara el propio portal web de consulta evitando tener más servidores para este propósito dado que el servir una única página web no es una tarea muy costosa. Esto quiere decir que cuando un cliente quiera acceder al portal web, este propio servicio le servirá la página web.

El software por el cual será implementado este servidor será con tecnología Java. Y además para ofrecer el servicio Rest se hará uso del Framework Jersey para servir las peticiones.

Este software podrá ser desplegado sobre un servidor Tomcat, Jboss o cualquier otro contenedor de aplicaciones java.

3.2.4 Portal web

Como se ha mencionado anteriormente, el propio servidor central será el encargado de ofrecer al cliente la página del portal web, como esta acción no requiere de una gran cantidad de información a transferir y no será muy excesiva, nos evitaremos de esta forma tener que añadir más servidores que suplan esta función.

La peculiaridad de este portal será la forma que tendrá de interactuar con los datos, esta peculiaridad se basa en que los datos no serán enviados a la vez que el portal cuando se solicite la página, sino que una vez que el portal haya sido enviado al cliente, este al procesar la página web del portal, realizara peticiones por medio de AJAX al servidor central por las cuales obtendrá los datos necesarios para representar las secciones. Estas peticiones serán realizadas únicamente cuando se cambie de sección en el portal y por defecto al inicio se cargará la sección de luminosidad. Desde la siguiente figura se puede ver de manera gráfica como interactuaran los diferentes componentes.

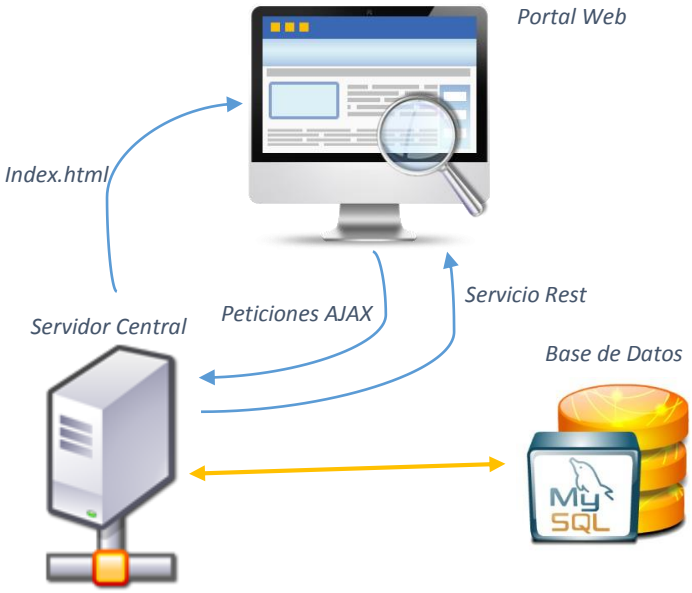


Fig. 3.2.13 Portal Web y Servidor

De esta manera será posible replicar el portal web en otro servidor de manera sencilla con lo que se permitirá balancear la carga en el caso de que existan una gran cantidad de clientes que quieren acceder al portal. En la figura inferior es posible ver la idea comentada de manera gráfica Todo ello se conseguirá de manera sencilla con la única modificación de la dirección de destino a la que tendrá que dirigirse para realizar las consultas.

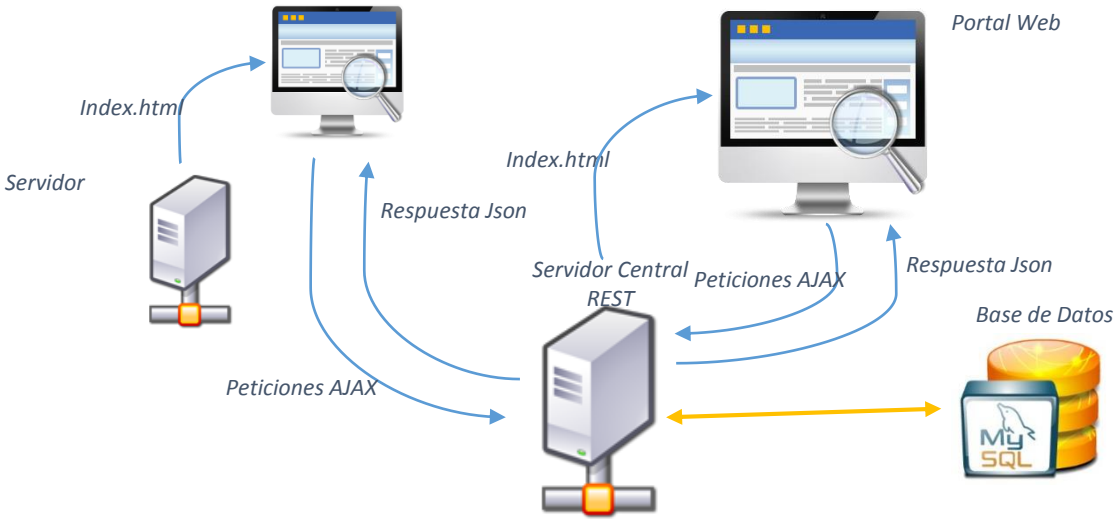


Fig. 3.2.14 Replicación del portal

El portal web estará basado en HTML, CSS y AngularJS para la lógica de la aplicación. Esté visualizará un resumen de la información (en la figura inferior es posible ver como ira estructurado el portal) almacenada en la base de datos de manera que podamos ver con un simple vistazo y de forma sencilla todos los datos referentes al panel solar.

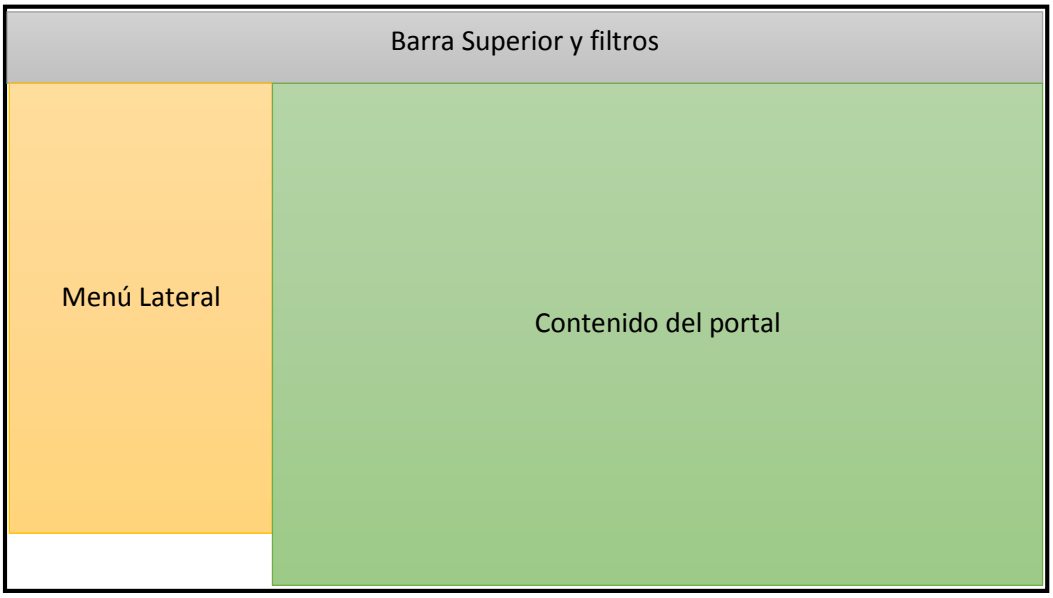


Fig. 3.2.15 Boceto del diseño del contenido del portal web

3.2.5 Base de datos

Para la tarea de persistencia de datos utilizaremos una base de datos MySQL, en la cual se creará una tabla para cada tipo de dato que se quiere almacenar. Esto es debido a la propia naturaleza de cada sensor los cuales proporcionan diferentes tipos de datos y si utilizáramos una única tabla perderíamos eficiencia debido a que tendríamos munchos campos de la tabla vacíos haciendo que dichas tablas ocupen mayor espacio y desaprovechando mucho almacenamiento, por este motivo finalmente se dispondrá de un total de 8 tablas, en donde se podrán diferenciar 6 de ellas las cuales serán utilizadas para el almacenamiento de los datos proporcionados por cada sensor y 2 tablas extra que serán empleadas para almacenar los eventos de error y estado que se registren en el panel con el objetivo de que puedan servir para llevar un pequeño control sobre el sistema.

En la siguiente figura se detallaran las tablas necesarias para almacear la información recibida con sus correspondientes atributos.

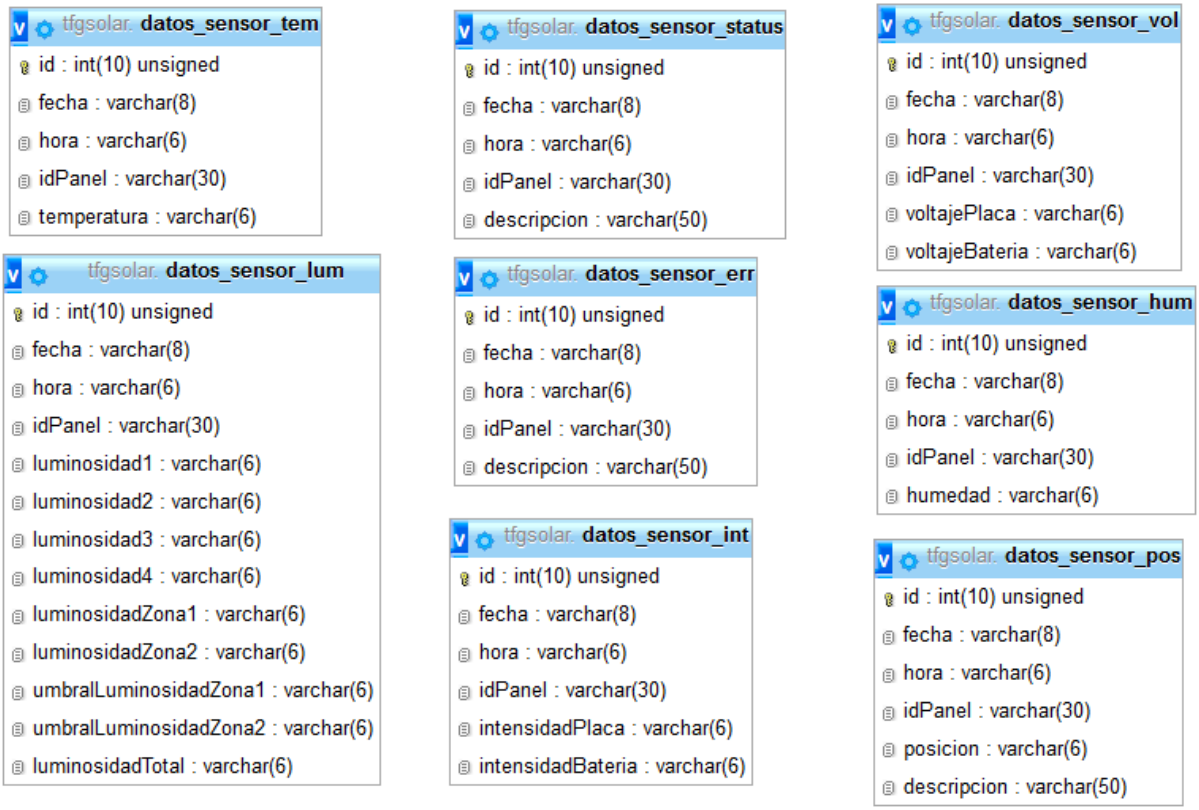


Fig. 3.2.16 Relación de tablas empleadas

Para realizar pruebas mientras se está desarrollando el portal web será necesario cargar la base de datos con datos de pruebas parecidos a los que serán enviados por el panel solar. Para ello se desarrollará unos procedimientos (se pueden ver los procedimientos en la figura inferior) por los cuales seremos capaces de generar estos datos de prueba y de esta manera poder hacer pruebas para realizar las comprobaciones oportunas en el portal. Por ejemplo, los datos sean visualizados en el formato correcto en el portal web o los gráficos muestren información útil.

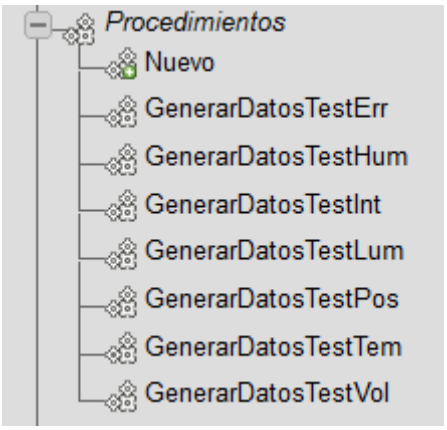


Fig. 3.2.17 Relación de procedimientos empleados para generar datos

4. Implementación

En este punto van a ser expuestos los pasos necesarios que se han tenido que seguir para realizar la implementación de todos los componentes hardware como software necesarios para el trabajo. Dado que el trabajo puede tener una pequeña base de IoT es necesario construir un hardware el cual sea capaz de obtener los datos del entorno que posteriormente se procesaran con el software desarrollado.

4.1 Hardware

Para este trabajo se ha llegado a implementar un pequeño prototipo de seguidor solar, para ello ha sido necesario contar con piezas hardware como el propio panel sobre el cual se va a aplicar la solución, así como materiales básicos para construir la plataforma del panel solar, además de necesitar dispositivos electrónicos por los cuales, gracias a la función que desempeñan son una parte muy importante en todo el conjunto.

4.1.1 Montaje plataforma móvil del panel solar

Debido a la problemática de necesitar ofrecer una inclinación al panel solar respecto al sol, será necesario construir un soporte el cual nos permita inclinarlo aprovecha así la mayor incidencia del sol posible.

Partimos de que tenemos un panel solar de pequeñas dimensiones (como la mostrada en la figura 4.1.1) sobre el cual se aplicara el prototipo. Este panel ofrece una tensión de salida de 18 Voltios, correcto para cargar una batería de 12 voltios y proporciona una corriente de 0.27 en su máxima potencia, llegando a ser capaz de producir 5W en las mejores condiciones posibles.

Fig. 4.1.1 Panel solar

Primero empezaremos buscando una superficie sobre la cual podamos trabajar en el soporte. Para esto emplearemos una caja de registro estanca como la mostrada en la figura inferior, la cual nos ofrecerá algunas ventajas respecto a otras alternativas.

Fig. 4.1.2 Caja de registro

Por un lado, al ser de una material robusto, nos servirá de apoyo para el soporte del panel el cual irá ubicado en la parte superior de la caja por medio, además de que también nos servirá

para alojar en el interior de la caja todos los dispositivos electrónicos necesarios para el control, obteniendo una protección dado que al ser estanca, protegerá todo el interior de las inclemencias meteorológicas.

Para conseguir hacer la inclinación del panel, se usara dos escuadras metálicas triangulares como la mostrada en la figura 4.1.3 las cuales estarán fijadas a la caja estanca, en el orificio superior de la escuadra se insertara una barra metálica con tuercas la cual hará la función de eje sobre la cual podrá rotar el soporte plano del panel.

Fig. 4.1.3 Escuadra para el eje central

Para hacer este soporte plano, se empleará una chapa de madera a la cual se le añadirá un marco en los márgenes, sobre los cuales descansará el panel y así evitaremos que este se mueva quedando fijo durante la inclinación del panel. Para permitir el movimiento de rotación, se le añadirá unos pequeños soportes los cuales harán la función de anclarlo al eje anteriormente añadido a las escuadras de la caja estanca, permitiendo que el panel este unido al soporte, además de permitiéndole la libertad de poder rotar sobre su propio eje.

Para conseguir la inclinación del soporte plano del panel solar unido a la caja por el eje, se añadirá un servomotor anclado a las escuadras de la caja, y se conectara al panel por medio de una varilla la cual permitirá transferir el movimiento del servomotor al soporte plano y de esta manera lograremos que este pueda inclinarse. A continuación en la figura de abajo se puede ver cómo va montado el servomotor a la estructura y como se une al soporte de la placa solar.

Fig. 4.1.4 Ubicación Servomotor

Hasta aquí se ha completado la parte de otorgar movilidad al panel solar consiguiendo que pueda ser inclinado cierto ángulo con el objetivo de orientarse lo más directo posible al sol.

4.1.2 Montaje del sistema de monitorización del panel solar

Para que todo lo realizado en el punto anterior tenga una funcionalidad, es necesario añadir dispositivos electrónicos que sean capaces de monitorizar su entorno y responder ante cualquier cambio producido o detectado. De esta manera podríamos desglosar este desarrollo en tres partes que forman todo el conjunto y funcionando todo al unísono conseguiremos obtener el resultado esperado.

Regulador de carga

Para este prototipo no va a ser necesario añadir un controlador de carga como tal porque el panel solar proporcionara como máximo 5w y no es necesario, pero sí que se va a necesitar un pequeño circuito por el cual nos permita obtener del panel solar una tensión estabilizada. Para ello lo que se hizo fue añadir un puente rectificador de diodos para evitar que la corriente fluya en sentido inverso cuando el panel tenga menor diferencia de potencial que la batería. Además se añadieron dos condensadores los cuales se encargan de mantener una tensión estable evitando picos producidos por el panel solar y que fueron observados durante las pruebas. A continuación se puede observar el esquema del circuito resultante de lo comentado arriba.

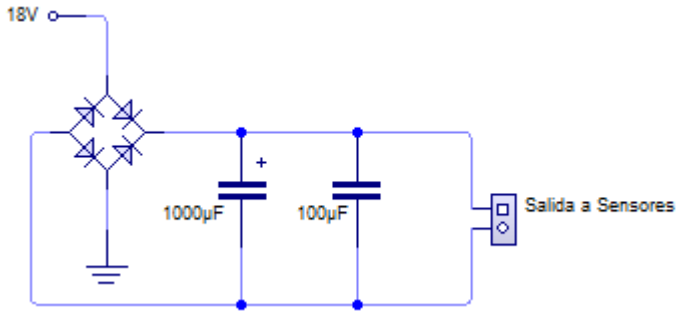


Fig. 4.1.5 Estabilizador de carga

Con este circuito conseguimos obtener una protección de las células fotovoltaicas del panel solar evitando que se dañen si estas producen menos voltaje que la batería. A la vez también nos proporcionara un voltaje estable continuo cuando el panel solar este generado energía dado que puede llegar a producir picos por encima de los 18 voltios. Finalmente el resultado del circuito puede observarse en la siguiente figura.

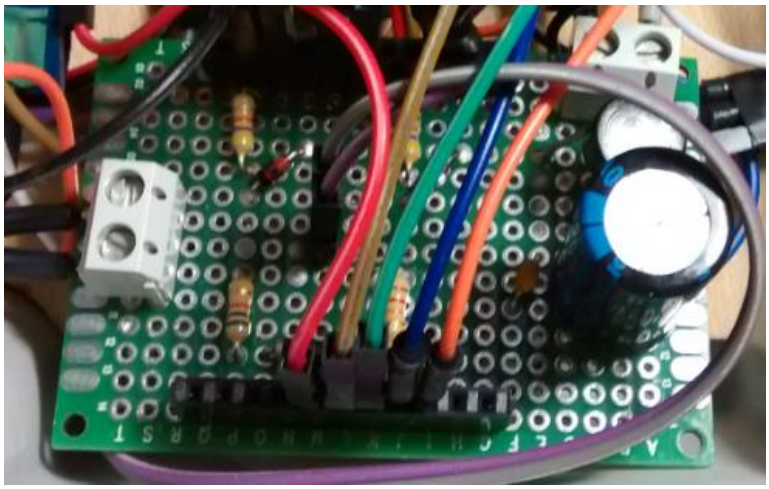


Fig. 4.1.6 Circuito estabilizador implementado

Sensores

Para hacer la monitorización del panel es necesario que se integren diferentes sensores para capturar las magnitudes físicas del entorno necesarias para llevar a cabo las tareas de control. Los sensores añadidos al trabajo van a ser de luminosidad, temperatura, humedad, voltaje e intensidad.

- Sensores de luminosidad

Para obtener la luminosidad radiada por el sol, haremos uso de 4 resistencias LDR, estas serán distribuidas de manera que dos serán colocadas en la parte superior del panel y otras dos serán colocadas en la parte inferior (en la figura 4.1.7 puede verse el esquema de conexionado y en la siguiente puede verse el sensor finalmente montado). Para ello será necesario hacer dos pequeñas placas de circuito impreso para montar en ella los componentes necesarios para construir el sensor. La razón de colocar dos resistencias LDR juntas es por el simple motivo de obtener una lectura más precisa, de esta forma al tener dos valores de cada resistencia, al hacer la media de ambos debería de salir un valor cercano al que ellos obtienen por separado, así también si uno de ellos fallara, aun podríamos seguir teniendo lecturas de valores por uno de ellos.

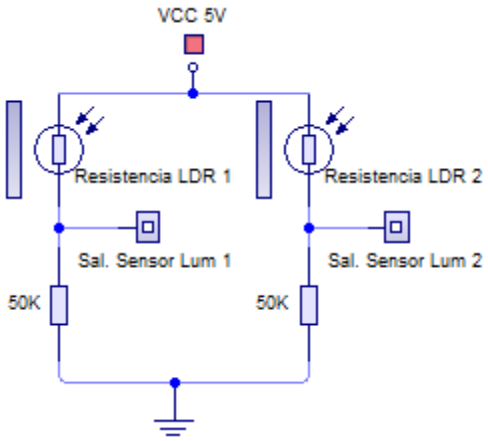


Fig. 4.1.7 Diseño del esquema sensor de luminosidad

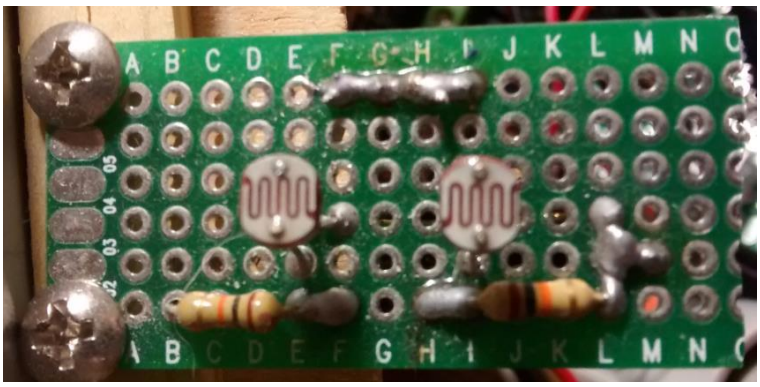


Fig. 4.1.8 Circuito impreso con los sensores de luminosidad

Para evitar que estos sensores estén consumiendo energía durante el periodo que no se está obteniendo valores de ellos, se ha optado por activar estos sensores por medio de un pin de la placa controladora Arduino en el momento anterior de leer sus valores, dejándole un cierto margen temporal para que el sensor sea capaz de capturar un valor correcto y una vez el micro-controlador ha obtenido el valor de los sensores, se vuelven a desconectar los sensores

consiguiendo un ahorro de energía porque solo son usados cuando es necesario, algo muy importante si en el futuro este dispositivo dependiera de la energía generada por el propio panel solar. Por medio de la siguiente figura se puede apreciar el esquema de conexionado.

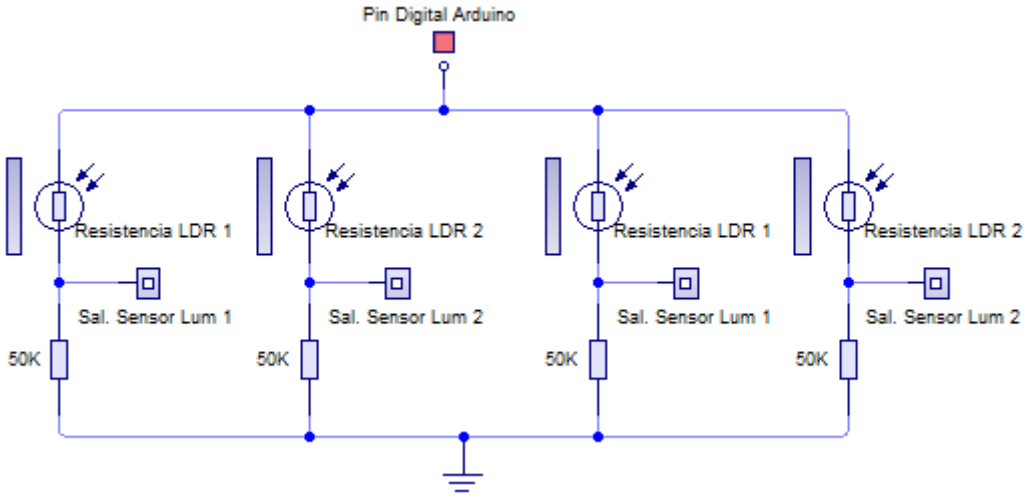


Fig. 4.1.9 Esquema circuito sensores luminosidad

Todas las salidas de estos sensores irán conectados a los pines de las entradas analógicas del micro-controlador dado que el rango de medida es un voltaje que va de 0 a 5V y es la única manera de que estos valores sean interpretados por el micro-controlador el cual nos ofrecerá un rango de 0 a 1024 de resolución de medida. El único inconveniente está en que hacemos uso de 4 pines analógicos del micro-controlador siendo estos algo escasos en otros micro-controladores.

- Sensor de temperatura y humedad

Como se ha comentado anteriormente, la temperatura es un factor clave para los paneles solares dado que a mayor temperatura sobre el panel, menor es la capacidad de generar energía que tiene. Por ese motivo se ha añadido un sensor de temperatura mediante el cual mediremos las temperaturas a las cuales está sometido y de esta manera podremos actuar en consecuencia.

El sensor empleado para esto va a ser el modelo DHT11, muy conocido en internet debido a su facilidad de uso, su bajo precio y además ofrece también la posibilidad de obtener la humedad del ambiente. Este sensor ira colocado en el centro sobre el soporte plano donde ira alojado el panel solar, de esta manera obtendremos la temperatura y humedad a la cual está sometido el panel. Este sensor es un módulo ya encapsulado que tiene tres patillas para su conexión a la placa Arduino. Dos de ellas corresponden a la alimentación del sensor (5v y masa) y la restante corresponde a la línea de datos. Esta línea de datos hace uso de un protocolo especial llamado 1-Wire por el cual envía la información al micro-controlador.

Para esta comunicación se conectara el sensor de temperatura DHT11 al pin digital 3 de la placa de desarrollo Arduino, este puede verse en la siguiente figura.

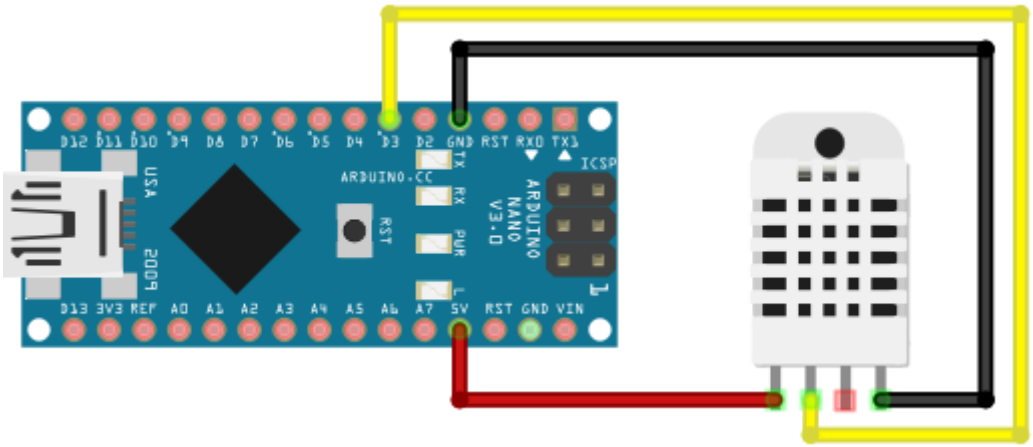


Fig. 4.1.10 Diagrama conexión sensor temperatura

- Sensores de voltaje

Dado que el panel puede proporcionar diferentes voltajes debido al principio de funcionamiento del propio panel, en donde a mayor luminosidad recibida en el panel, mayor voltaje de salida tendrá, será interesante añadir sensores que nos monitoricen estos valores para tener un control sobre que voltaje genera el panel y que voltaje tiene la batería. Para ello se creara un circuito por el cual obtendremos el voltaje existente generado por el panel y la batería. Este circuito se denomina divisor resistivo, y con el podemos reducir el voltaje al valor de referencia del cual podremos obtener el valor a medir.

Como medida de protección, se ha añadido un diodo zener de 5,1 V a la salida del sensor para evitar que dicho circuito nunca de más de 5V en caso de aumentar el voltaje de entrada y pueda quemar el micro-controlador.

Se añadirán dos sensores de este tipo (puede verse en la figura inferior su esquema), uno para monitorizar el voltaje generado por el panel solar de forma que seremos capaces de ver las fluctuaciones que pueda tener y el otro se utilizara para monitorizar el voltaje almacenado en la batería. Estos sensores irán conectados a dos pines analógicos de la placa de desarrollo Arduino.

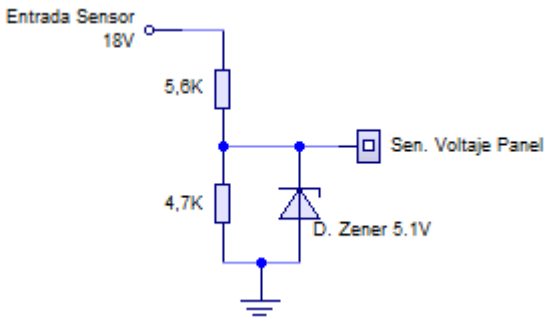


Fig. 4.1.11 Esquema divisor resistivo para el sensor voltaje

- Sensores de intensidad de corriente

Dado que es importante la potencia generada por el panel solar, necesitamos obtener que intensidad de corriente está generando y de este modo seremos capaces por ejemplo de predecir cuánto tiempo puede tardar en cargarse la batería, así como los máximos y mínimos de

generación que pueden existir. Para esta tarea se hará uso del sensor de corriente ACS712 con una tolerancia de 5A.

Este sensor se vende ya ensamblado así que su instalación resulta sencilla. Dispone de dos bornes con tornillo en un lado en los cuales se conecta el circuito a medir, y al otro lado tiene tres pines los cuales dos de ellos corresponden a la alimentación (5V y masa) y otro corresponde a la salida analógica de datos del valor obtenido. De esta forma el pin de datos ira conectado a un pin analógico de la placa de desarrollo Arduino. En la siguiente figura puede verse como se realiza la conexión entre los módulos.

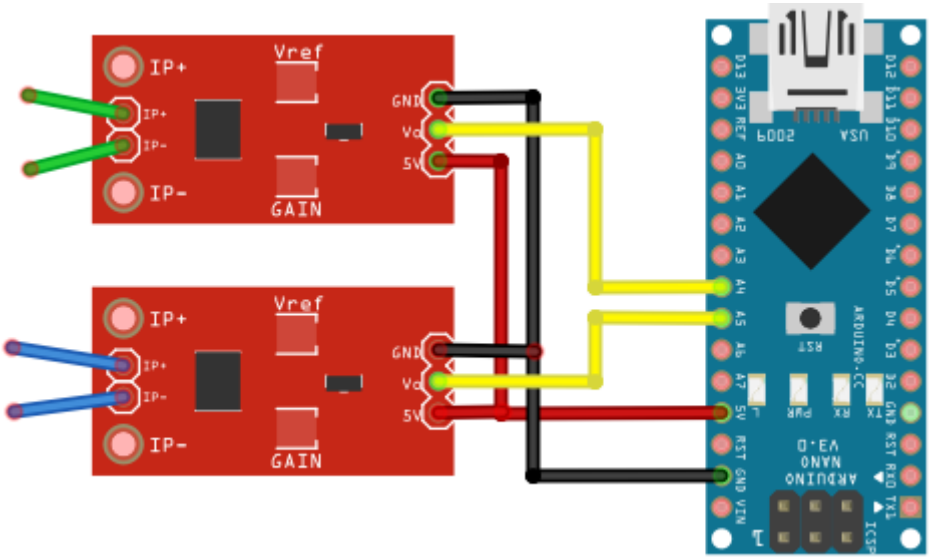


Fig. 4.1.12 Diagrama de conexión de los sensores de intensidad

A continuación en la siguiente figura se muestra el esquema del circuito implementado en el cual se puede apreciar cómo van acoplados en el circuito de regulación de carga tanto los sensores de voltaje como los de intensidad de corriente.

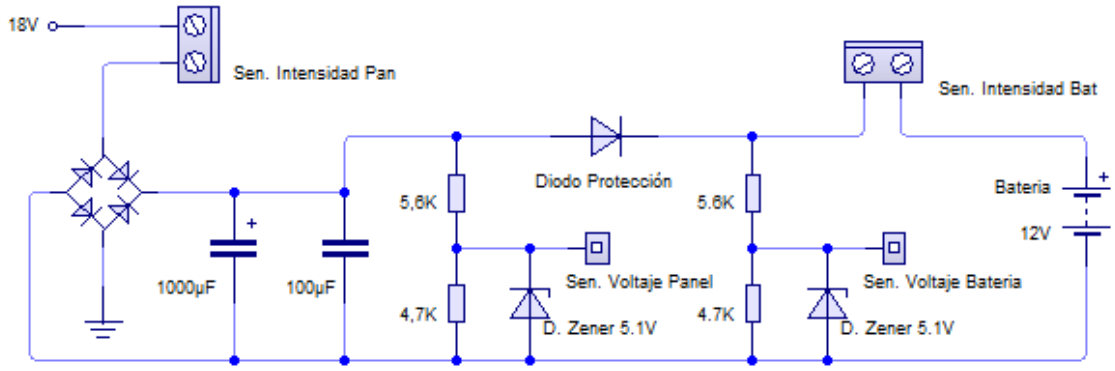


Fig. 4.1.13 Esquema circuito final

Controlador

Una vez tenemos instalados los sensores en el panel, necesitaremos de un dispositivo por el cual podamos obtener y procesar los datos de los sensores, para ello vamos a hacer uso de una placa de desarrollo Arduino Nano (figura 4.1.14) junto a un módulo de conexionado el cual nos facilitara la manera en la que podamos conectar los sensores a la placa de desarrollo (figura 4.1.15).

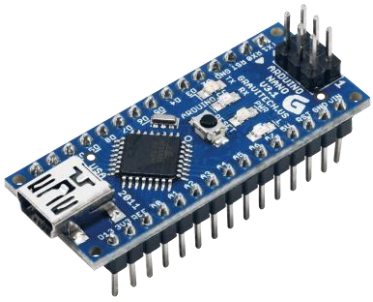


Fig. 4.1.14 Arduino Nano

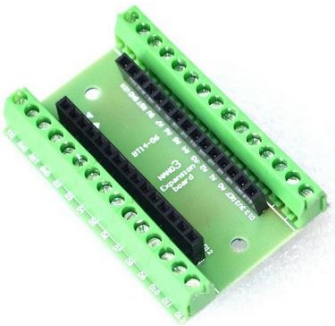


Fig. 4.1.15 Modulo de conexionado por tornillo

Esta placa de desarrollo al tener un micro-controlador será capaz de leer de los sensores los valores del entorno y actuar dependiendo lo que se le haya programado. Todo los pines analógicos de la placa de Arduino estarán usados por los sensores de luminosidad, voltaje e intensidad de corriente y únicamente haremos uso de tres pines digitales, uno para activar los sensores de luminosidad, otro para controlar el servomotor y otro para obtener los datos del sensor de temperatura. Sería posible usar otro pin digital para controlar un relé el cual desconectaría l batería del panel solar cuando esta esté totalmente cargada pero esto no será implementado en el prototipo.

- Comunicación

Para realizar la comunicación de la placa solar a la maquina pasarela por medio de la API software, se utilizará la propia interfaz USB que ofrece la placa de desarrollo conectándolo por medio de un cable USB de forma directa (figura 4.1.16), de esta forma la comunicación resulta sencilla porque al conectar la placa a la máquina, esta con su sistema operativo la identificara como un puerto Serial al cual podremos conectarnos y consultar los datos obtenidos.

Fig. 4.1.16 Conexión por USB

Existe también otra alternativa por la cual podemos conectar la placa de desarrollo a la maquina pasarela. Esta consiste en emplear una red de área local Ethernet en la cual conectaremos ambos dispositivos. Para ello será necesario dotar a la placa de desarrollo de un módulo Ethernet por el cual se podrá conectar a la red local (figura 4.1.17).

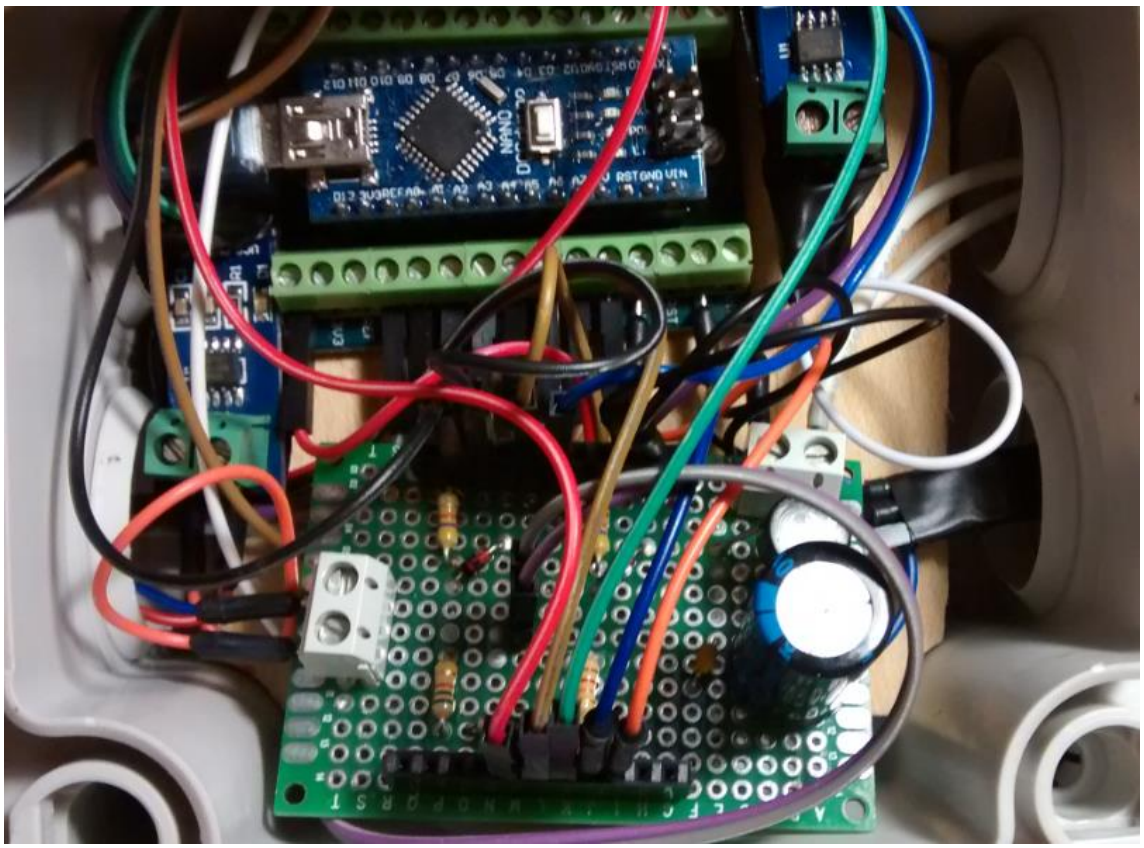
Fig. 4.1.17 Modulo Ethernet

Debido a la limitación de las anteriores conexiones de tener dependencia de estar conectado de forma física a la máquina, estuve probando realizar la comunicación de forma inalámbrica por

medio de un módulo WIFI. Este módulo corresponde al ESP8266 y la manera de comunicarse a la placa de desarrollo es a través de un puerto serial (figura 4.1.18). Finalmente no conseguí hacer una implementación estable de este módulo debido a la manera que tiene de configurarse y de conectarse a las redes inalámbricas. Todo esto se realiza por medio de comandos AT, y algunas veces estos comandos no eran aceptados de manera correcta o bien el modulo se quedaba congelando, obligando a reiniciar todo el conjunto.

Fig. 4.1.18 Modulo de comunicación WIFI

En la siguiente figura se muestra el resultado final de todo el hardware que ha sido necesario implementar en el interior de la caja de registro.



4.2 Software

4.2.1 Firmware Placa desarrollo Arduino

Para implementar el software que va alojado en la placa de desarrollo Arduino vamos a usar el entorno de desarrollo propio que proporciona la plataforma Arduino. Con este entorno se realizara el software necesario el cual será el encargado de obtener y procesar los datos de los sensores. Para implementar dicho código, este será organizado por ficheros (figura 4.2.0) de

manera que sea más fácil ordenar el código así como para que su mantenimiento y modificación sea más sencilla.

Fig. 4.2.0 Ficheros creados para ordenar la lógica

El fichero principal y el cual contiene el programa principal del sistema es el fichero llamado "tfg_panel_solar". En él se inicializaran las variables necesarias así como los sensores y actuadores del sistema desde la directiva setup(). En este fichero se declarará el número de muestras que serán tomadas por el sistema de monitorización y de las cuales posteriormente se obtendrá la media de los valores, así también el identificador que tendrá cada panel solar.

Cuando se empiece a ejecutar el programa principal por medio del loop, este empezara a obtener valores de los diferentes sensores e ira incrementando un contador de manera que cuando este contador alcance el número de muestras establecido, de esta manera el panel solar empezará a realizar un seguimiento de manera autónoma al sol, orientándolo de forma que obtengamos la máxima radiación producida por el sol. Además el panel dispone de un temporizador el cual se encargara de enviar los datos obtenidos por los sensores a la maquina pasarela dependiendo del tiempo especificado.

Por defecto está configurado para que cada 20 segundos envíe la información obtenida a la maquina remota, pero para una mejor optimización, el valor correcto debería de ser de 15 minutos.

También se realizara un chequeo del puerto serie para el caso de que se reciban comandos de configuración o consulta del panel solar y así sean tratados por el sistema. Si es un comando de consulta, este enviara un mensaje JSON con los valores solicitados.

En el fichero de comunicaciones estarán todos los métodos necesarios para realizar las comunicaciones. Para ello se han implementado un método principal llamado readPortSerial() el cual es invocado desde el programa principal y su función es la de identificar si se recibe algún mensaje por el puerto serie.

Existe una funcionalidad del propio entorno de desarrollo con la que también se puede implementar esta funcionalidad por medio de un evento, el cual es activado cuando se recibe información a través del puerto serie, pero este método no lo empleo porque este procedimiento solo está escuchando el puerto serie físico del micro-controlador y como el micro-controlador usado solo dispone de un puerto serie físico, al realizar las pruebas de conectividad era necesario hacer uso de un puerto serie software, así que fue necesario crear un procedimiento que hiciera la tarea de escuchar dicho puerto.

Cuando recibimos un mensaje por el puerto serie, es necesario identificarlo para saber qué tipo de solicitud se ha recibido y saber cómo se debe actuar. Para ello se implementara un método llamado "parserMensaje()" el cual recibirá un mensaje y se identificara que tipo de petición es y se actuara invocando el método que corresponda a la petición solicitada.

Los métodos creados para responder a las peticiones, envían la información por medio de mensajes JSON, a continuación se indican los métodos implementados para ello.

- getAll()

Este método invoca a los métodos getTem(), getHum(), getLuminosidad(ALL), getVol(), getInt(), getPos() para devolver la información de manera conjunta. Enviando cada uno de ellos el mensaje correspondiente.

- getTem()

  Este método devuelve el valor de temperatura obtenido por el sensor.

- getHum()

  Este método devuelve el valor de humedad obtenido por el sensor.

- getVol()

  Este método devuelve los valores de voltaje obtenido por el sensor situado en el panel y en la batería.

- getInt()

  Este método devuelve los valores de intensidad obtenido por el sensor situado en el panel y en la batería.

- getPos()

  Este método devuelve el valor de la posición en la que se encuentra el panel.

- getLuminosidad()

  Este método devuelve los valores de luminosidad obtenido por el sensor. Estos valores pueden ser:

  - Valor de cada sensor de luminosidad
  - Valor del conjunto de los sensores de arriba o abajo
  - Valor umbral del conjunto de los sensores de arriba o abajo
  - Valor total de todos los sensores de luminosidad

- setPos()

  Este método permite establecer la posición del panel solar a un valor que se especifique en un mensaje enviado.

- getIdPlaca()

  Este método permite obtener el identificar que tiene el panel solar.

- setIdPlaca()

  Este método permite establecer el identificador de cada placa. Este identificador se utilizara para diferenciar una placa de otra.

- getDesconocido()

  Este método devolverá un mensaje de error en caso de recibir un mensaje no reconocido por el método de parseo.

- sendMensaje()

  Este método es el encargado de enviar los mensajes generados por los métodos correspondientes a través del puerto serie.

Para el envío de la información se hace uso de mensajes JSON, a continuación se especifica los distintos tipos de mensajes enviados, todos ellos son diferenciados por la clave MSG en la que se le especifica el tipo de mensaje:

- getIdPlaca()
    - “ID:PLACA01”
- getTem()
    - "{ID:PLACA01,MSG:TEM,VAL:24.50}"
- getHum()
    - "{ ID:PLACA01,MSG:HUM,VAL:40.00}"
- getLuminosidad() → los mensajes de luminosidad se pueden diferenciar gracias a la clave SEN.
    - "{ID:PLACA01,MSG:LUM,SEN:1,VAL:400}"
    - "{ ID:PLACA01,MSG:LUM,SEN:2,VAL:400}"
    - "{ ID:PLACA01,MSG:LUM,SEN:3,VAL:400}"
    - "{ ID:PLACA01,MSG:LUM,SEN:4,VAL:400}"
    - "{ ID:PLACA01,MSG:LUM,ZON:A,VAL:400}"
    - "{ ID:PLACA01,MSG:LUM,ZON:B,VAL:400}"
    - "{ ID:PLACA01,MSG:LUM,UMB:1,VAL:400}"
    - "{ ID:PLACA01,MSG:LUM,UMB:2,VAL:400}"
    - "{ ID:PLACA01,MSG:LUM,VAL:400}"
- getVol() → En estos mensajes se envía tanto el valor del panel como de la batería a través de la clave VAL.
    - "{ ID:PLACA01,MSG:VOL,VAL:{PAN:4.5, BAT:5}}"
- getInt() → En estos mensajes se envía tanto el valor del panel como de la batería a través de la clave VAL.
    - "{ ID:PLACA01,MSG:INT,VAL:{PAN:0.10,BAT:0.10}}"
- getPos()
    - "{ ID:PLACA01,MSG:POSPAN,VAL:100}";
- getDesconocido()
    - "{ ID:PLACA01,MSG:ERR,DES:COMMAND_UNKNOWN}"

El fichero movimiento contiene las variables empleadas para la configuración del servo-motor, así como los valores mínimos y máximos ángulos permitidos del servomotor, también tendremos una variable que nos indicará el ángulo en el que se encuentre el panel solar. El método para realizar el seguimiento del sol, se llama “orientarPanelSolar()” y este básicamente consistirá en obtener los valores de los sensores por medio de dos procedimientos los cuales obtienen el valor medio del número total de muestras obtenidas.

De esta manera lo que se hace es comprobar de las zonas de luminosidad (superior e inferior) sus valores de manera que si en la zona superior tiene un valor mayor que en el inferior, entonces el panel subirá orientándolo hacia arriba, y si en la zona inferior tenemos mayor luminosidad que en la zona superior, entonces se bajara el panel.

Cabe mencionar que cada vez que se inicializa el programa inicial, se invoca al método "inicializarServoInclinacion()" el cual se encarga de indicar en que pin de salida del micro-controlador tenemos conectado el servo-motor y establece el ángulo del panel a 0.

En el fichero sensores tendremos todos los métodos relacionados con la obtención de los datos de los sensores. Aquí se inicializaran las variables de los pines necesitados para los sensores, así como las variables en las cuales se almacenaran los datos obtenidos. Para realizar la captura de los datos se implementaran los siguientes métodos:

- obtenerDatosIntensidad()

  Con este método seremos capaces de obtener los valores proporcionados por los pines analógicos a los que están conectados el sensor de corriente del panel y de la batería.

- obtenerDatosVoltaje()

  Con este método seremos capaces de obtener los valores proporcionados por los pines analógicos a los que están conectados el sensor de voltaje del panel y de la batería.

- obtenerDatosLuminosidad()

  Con este método seremos capaces de obtener los valores proporcionados por los pines analógicos a los que están conectados los sensores de luminosidad.

- obtenerDatosTempHum()

  Con este método seremos capaces de obtener los valores proporcionados de temperatura y humedad por el sensor, para ello se usara la librería DHT.h la cual nos ayuda a gestionar la manera de cómo usar el protocolo 1-wire del que hace uso este tipo de sensor.

Además de los métodos de captura tenemos también otros que nos permitirán controlar los sensores.

- inicializarSensorLuminosidad()

  Con este método seremos capaces de activar los sensores de luminosidad con el objetivo de ahorra energía, y solo serán activados en el momento que vayan a ser empleados.

- inicializarSensorTemHum()

  Con este método inicializamos el objeto que gestiona el sensor de temperatura.

También tendremos métodos auxiliares que nos ayudaran a realizar los cálculos medios de los valores a la hora de consultarlos. De esta manera cuando se reciba un tipo de mensaje de consulta, se invocara a unos de estos métodos para que calcule su valor medio.

Se creó un fichero Wifi el cual contiene los métodos necesarios para crean una comunicación inalámbrica por medio del módulo ESP8266. Como ya mencioné más arriba, esta funcionalidad no llego a implementarse completamente, debido a que es inestable, y a veces provoca que se

requiera reiniciar el sistema. Este fichero tendrá los métodos necesarios para establecer la comunicación con el propio módulo ESP8266, debido a que este módulo hace uso de comandos AT para llevar a cabo toda la comunicación, fue necesario crear unos métodos por los cuales podamos realizar todas las acciones de manera rápida y sencilla.

4.2.2 API de comunicaciones hardware – software

Para la implementación de esta API de comunicaciones, se creara un proyecto java en el entorno de desarrollo Eclipse. Este deberá ser configurado de manera que se haga uso del SDK de java en la versión 8.

Como será necesario hacer uso de otras librerías, en vez de buscarlas por internet, descargarlas e importarlas al proyecto, lo que utilizaremos para esta tarea será Maven. Esto ayudara a gestionar las librerías necesarias así como las dependencias que necesiten. Para esto es necesario convertir el proyecto java normal a un proyecto Maven. Para ello debemos seleccionar el proyecto y al pulsar el botón derecho, aparecerá el menú en donde iremos a Configure y pulsaremos sobre "Convert to Maven Proyect" (figura 4.2.1). De esta manera se adaptara el proyecto java a un proyecto Maven.

Fig. 4.2.1 Convertir proyecto a Maven

Una vez convertido el proyecto tendremos que editar el fichero encargado de configurar Maven. Este fichero se llama pom.xml y contendrá todas las librerías que necesitaremos para la implementación.

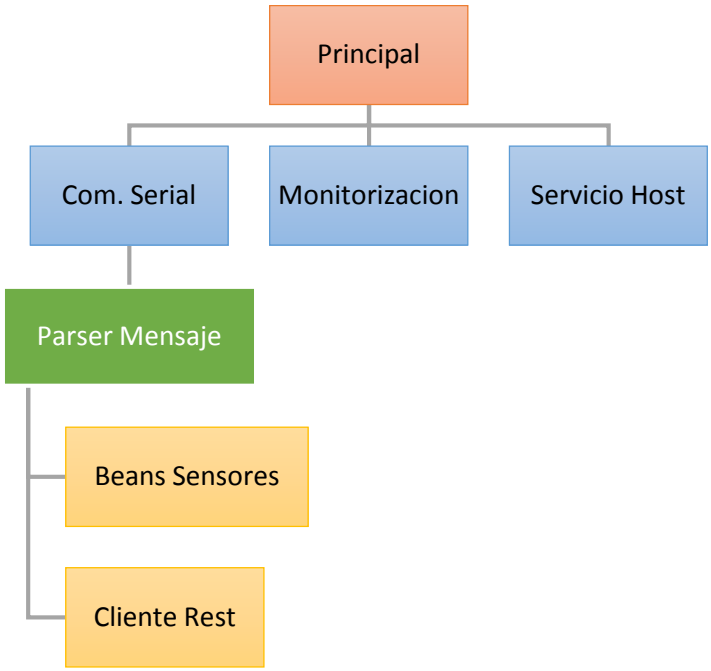
```
<dependencies>

	<dependency>
		<groupId>org.rxtx</groupId>
		<artifactId>rxtx</artifactId>
		<version>2.1.7</version>
	</dependency>

	<dependency>
		<groupId>org.json</groupId>
		<artifactId>json</artifactId>
		<version>20151123</version>
	</dependency>

	<dependency>
		<groupId>com.sun.jersey</groupId>
		<artifactId>jersey-client</artifactId>
		<version>1.19</version>
	</dependency>

</dependencies>
```

También desde este fichero se especifica de qué manera se construirá el paquete final el cual será ejecutado y que contendrá los ficheros compilados de la aplicación así como los ficheros extra que serán importados al paquete de exportación.

```
<build>
	<sourceDirectory>src</sourceDirectory>

	<resources>
		<resource>
			<directory>${basedir}/lib_rx/</directory>
			<filtering>false</filtering>
			<includes>
				<include>*.*</include>
			</includes>
		</resource>
	</resources>

	<plugins>
		<plugin>
			<artifactId>maven-compiler-plugin</artifactId>
			<version>3.3</version>
			<configuration>
				<source>1.8</source>
				<target>1.8</target>
			</configuration>
		</plugin>

		<plugin>
			<groupId>org.apache.maven.plugins</groupId>
			<artifactId>maven-jar-plugin</artifactId>
			<configuration>
				<archive>
					<addMavenDescriptor>false</addMavenDes
				criptor>
					<compress>true</compress>
					<manifest>
					<mainClass>es.upv.Inicio</mainClass>
					<addClasspath>true</addClasspath>
					<classpathPrefix>../lib/</classpathPre
				fix>
					</manifest>
				</archive>
			</configuration>
		</plugin>

		<plugin>
			<groupId>org.apache.maven.plugins</groupId>
			<artifactId>maven-dependency-plugin</artifactId>
			<version>2.5.1</version>
			<executions>
			  <execution>
				<id>copy-dependencies</id>
				<phase>package</phase>
				<goals>
					<goal>copy-dependencies</goal>
				</goals>
				<configuration>
					<includeScope>runtime</includeScope>
					<outputDirectory>${project.build.direc
```

```
                                    tory}/dependency-
                                    jars/</outputDirectory>
                            </configuration>
                        </execution>
                    </executions>
                </plugin>

                <plugin>
                    <artifactId>maven-assembly-plugin</artifactId>
                    <configuration>
                        <archive>
                            <manifest>
                                <mainClass>es.upv.Inicio</mainCl
                            ass>
                            </manifest>
                        </archive>
                        <descriptorRefs>
                            <descriptorRef>jar-with-
                        dependencies</descriptorRef>
                        </descriptorRefs>
                    </configuration>
                </plugin>
            </plugins>
        </build>
```

Una vez configurado el entorno de desarrollo y descargadas las librerías necesarias, empezaremos con la tarea de implementar el código. Para ello se creara la siguiente estructura para organizar el código.



La **clase Principal**, llamada Inicio.java será el programa de inicio de la aplicación dado que este contendrá el método main(). En un principio se pensó en añadir una opción de poder pasarle un parámetro a la hora de lanzar el programa, este era un modo debug el cual nos permitiría poder generar información simulando un panel solar. Finalmente no fue necesario dado que se hizo un procedimiento por base de datos el cual nos suplió esta funcionalidad. Sí que se ha implementado la opción de poder poner un parámetro para indicar la dirección del servidor al cual se debe de enviar la información obtenida.

Desde esta clase se inicializará las clases de Monitorización y ServicioHost. Para la clase de Monitorización será necesario obtener el puerto serie que esté conectado a la máquina y que queremos monitorizar.

La **clase monitorización** se realizó desde el principio con la idea de ser la encargada de poder ejecutar eventos programados para tareas de consulta de información o modificación de parámetros en los dispositivos hardware de los paneles solares. Finalmente se descartó esta idea debido a que es más práctico que sean los propios dispositivos lo que envíen la información cada cierto tiempo, liberando de carga a este software dado que es capaz de recibir información de diversas fuentes.

Esta clase realiza la conexión al puerto serie que ha detectado en el sistema a una velocidad de 57600 bps. Esta clase tiene una limitación y es que solo es posible abrir una única conexión por un puerto serie. En el caso de que existieran más dispositivos conectados por puerto serie a la máquina y se quisiera monitorizarlos, se debería realizar un pequeña modificación en el código.

La **clase CommSerial** es la encargada de realizar todas las conexiones a los puertos serie de la API. Desde esta clase seremos capaces de buscar los puertos serie que hay en el sistema y en el caso de que existan, se obtendrá en este caso el primero de ellos. Este apartado es que he comentado más arriba donde existe una limitación en la que únicamente podemos monitorizar un puerto serie a la vez. Para estas tareas se han implementado los siguientes métodos:

```
public boolean buscarPuertos() {
	boolean existenPuertoSerial = false;
	ports = CommPortIdentifier.getPortIdentifiers();

	System.out.println("Listado puertos: ");
	while (ports.hasMoreElements()) {
		CommPortIdentifier curPort =
(CommPortIdentifier)ports.nextElement();

		if (curPort.getPortType() == CommPortIdentifier.PORT_SERIAL) {
			System.out.println("- " + curPort.getName());
			portMap.put(curPort.getName(), curPort);
			existenPuertoSerial = true;
		}
	}

	return existenPuertoSerial;
}

public String obtenerPuerto() {
	if (!portMap.isEmpty()) {
		Collection listadoPuertos = portMap.values();
		Iterator <CommPortIdentifier> iterador =
listadoPuertos.iterator();
		CommPortIdentifier puerto = iterador.next();
		return puerto.getName();
	}

	return "";
}
```

Dado que son los propios dispositivos los que envían la información cada cierto tiempo, era necesario implementar un evento que estuviera escuchando el puerto serie por si se recibe información a través de él. La propia API de la librería del puerto serie nos proporciona el poder implementar este evento de manera sencilla. Para ello sobrescribimos el método llamado

serialEvent() el cual estará atento a recibir cualquier evento que provenga de un puerto serie, de esta manera cuando recibamos cualquier mensaje, este será tratado por medio de una clase especialmente creada para esta tarea.

```
public void serialEvent(SerialPortEvent evento) {
	if (evento.getEventType() == SerialPortEvent.DATA_AVAILABLE) {
		try {
			char singleData = (char)input.read();
			if (singleData != NEW_LINE_ASCII) {
				mensaje += singleData;
			} else {
				tratarMensaje.tratarMensaje(mensaje);
				mensaje = "";
			}
		} catch(Exception e) {
			System.out.println(e.getMessage());
			System.exit(0);
		}
	}
}
```

La **clase ParserMessage** es la encarga de interpretar los mensajes recibidos por medio del puerto serie o también por el servicio de red. Esta clase es de vital importancia dado que es la encargada de procesar y enviar la información recibida al servidor central por medio de un servicio REST.

Para la tarea de procesar los mensajes recibidos se han creado diferentes JavaBeans (figura 4.2.2) los cuales son objetos encargados de almacenar la información de estos mensajes de manera organizada dependiendo el tipo de mensaje recibido.

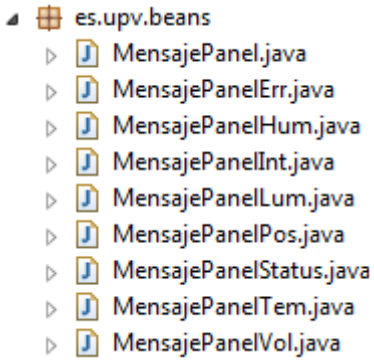


Fig. 4.2.2 Listado de objetos beans

Para llevar a cabo esto, se debe convertir el mensaje recibo en formato string a un objeto JSON del cual podamos obtener la información de manera sencilla. Para esta tarea se ha empleado la librería JSON de java la cual nos permite a partir de un mensaje plano de texto, generar un objeto JSON del cual podemos extraer la información en su formato correcto y de una manera sencilla.

De esta forma se puede generar un objeto JSON del cual podemos extraer el tipo de mensaje que es y poder tratarlo dependiendo su tipo. Para esto lo que se hizo fue crea un objeto bean genérico del cual heredan el resto de objetos específicos, de esta manera es posible generar un objeto dependiendo del tipo de mensaje recibo y guardarlo en un objeto genérico.

```
public void tratarMensaje(String mensaje) {
	MensajePanel objPanelGenerico = null;
	try {
		JSONObject obj = new JSONObject(mensaje);
		String idPlaca = obj.getString("ID");
```

```
            if (!obj.isNull("MSG")) {
                    String msg = obj.getString("MSG");
                    if ("TEM".equals(msg)) {
                            MensajePanelTem objTemp = new MensajePanelTem();
                            objTemp.setIdPanel(idPlaca);
                    objTemp.setValor(String.valueOf((obj.getDouble("VAL"))));
                            objPanelGenerico = objTemp;
                    } else if ("HUM".equals(msg)) {
                            MensajePanelHum objHum = new MensajePanelHum();
                            objHum.setIdPanel(idPlaca);
                    objHum.setValor(String.valueOf((obj.getDouble("VAL"))));
                            objPanelGenerico = objHum;
                    } else if ("VOL".equals(msg)) {
                            MensajePanelVol objVol = new MensajePanelVol();
                            objVol.setIdPanel(idPlaca);
        objVol.setValorBat(String.valueOf(obj.getJSONObject("VAL").getDouble("BAT")));
        objVol.setValorPan(String.valueOf(obj.getJSONObject("VAL").getDouble("PAN")));
                            objPanelGenerico = objVol;
                    } else if ("INT".equals(msg)) {
                            MensajePanelInt objInt = new MensajePanelInt();
                            objInt.setIdPanel(idPlaca);
        objInt.setValorBat(String.valueOf(obj.getJSONObject("VAL").getDouble("BAT")));
        objInt.setValorPan(String.valueOf(obj.getJSONObject("VAL").getDouble("PAN")));
                            objPanelGenerico = objInt;
                    } else if ("POSPAN".equals(msg)) {
                            MensajePanelPos objPos = new MensajePanelPos();
                            objPos.setIdPanel(idPlaca);
                            if (!obj.isNull("VAL")) {
                            objPos.setValor(String.valueOf(obj.getInt("VAL")));
                            }
                            if (!obj.isNull("DES")) {
            objPos.setDescripcion(String.valueOf(obj.getString("DES")));
                            } else if (!obj.isNull("ERR")) {
            objPos.setDescripcion(String.valueOf(obj.getString("ERR")));
                            }
                            objPanelGenerico = objPos;
                    } else if ("LUM".equals(msg)) {
                            MensajePanelLum objLum = new MensajePanelLum();
                            objLum.setIdPanel(idPlaca);
        objLum.setValorS1(String.valueOf(obj.getJSONObject("VAL").getInt("S1")));
        objLum.setValorS2(String.valueOf(obj.getJSONObject("VAL").getInt("S2")));
        objLum.setValorS3(String.valueOf(obj.getJSONObject("VAL").getInt("S3")));
        objLum.setValorS4(String.valueOf(obj.getJSONObject("VAL").getInt("S4")));
        objLum.setValorZonaA(String.valueOf(obj.getJSONObject("VAL").getInt("ZA")));
        objLum.setValorZonaB(String.valueOf(obj.getJSONObject("VAL").getInt("ZB")));
        objLum.setValorUmbralA(String.valueOf(obj.getJSONObject("VAL").getInt("U1")));
        objLum.setValorUmbralB(String.valueOf(obj.getJSONObject("VAL").getInt("U2")));
        objLum.setValorTotal(String.valueOf(obj.getJSONObject("VAL").getInt("TOT")));
                            objPanelGenerico = objLum;
                    } else if ("ERR".equals(msg)) {
                            MensajePanelErr objErr = new MensajePanelErr();
                            objErr.setIdPanel(idPlaca);
                            objErr.setDescripcion(obj.getString("DES"));
                            objPanelGenerico = objErr;
                    } else if ("STA".equals(msg)) {
                            MensajePanelStatus objStatus = new MensajePanelStatus();
                            objStatus.setIdPanel(idPlaca);
                            objStatus.setDescripcion(obj.getString("DES"));
                            objPanelGenerico = objStatus;
                    }
            }

            if (objPanelGenerico != null) {
                    enviarMensajeServidor(objPanelGenerico);
            }
    } catch (JSONException ex) {
            System.out.println("Error: " + ex.getMessage());
```

```
        } catch (Exception ex1) {
                System.out.println("Error: " + ex1.getMessage());
        }
}
```

Del procedimiento tratarMensaje(), existe un paso en el cual enviamos el objeto genérico por medio del procedimiento enviarMensajeServidor() siempre y cuando el objeto genérico esté creado.

Desde este procedimiento volvemos a generar un objeto JSON a partir del objeto bean genérico. Visto de esta manera el paso anterior en el cual realizamos una conversión puede resultar redundante, pero podría servir para validar la información recibida o bien para implementar una pequeña cola de mensajería en la cual se almacenarían los mensajes hasta cierto criterio para posteriormente enviarlos de nuevo. Y finalmente invocamos desde el propio procedimiento a un servicio REST el cual enviara los mensajes a un servidor central.

```
private void enviarMensajeServidor(MensajePanel mensaje) {

        try {
                JSONObject obj = new JSONObject(mensaje);
                ClienteRest servicioClienterRest = new
        ClienteRest(dirservicio,obj.toString());
                servicioClienterRest.enviarMensaje();
        } catch (JSONException ex) {
                System.out.println(ex.getMessage());
        }
}
```

Para realizar la comunicación al servicio REST se ha implementado la **clase ClienteRest**. Esta tendrá la lógica para conectarse a una dirección web correspondiente a un servidor y enviara la información por medio de HTTP vía POST a dicho servidor. A esta clase es posible establecerle la dirección en la que se encuentra el servidor al que se tendrá que conectar el servicio. Para ello es necesario añadir el parámetro IPSERVICIO=192.168.1.X justo cuando se lance el comando para ejecutar el fichero generado .jar. Cuando se invoque el procedimiento enviarMensaje() este realizara la tarea anteriormente mencionada. Se conectara al servidor y enviara la información.

```
public void enviarMensaje() {
        try {
                Client client = Client.create();
                WebResource webResource = client.resource(dirRecursoServidor);

                ClientResponse response = webResource.type(tipoMensajeJson)
                   .post(ClientResponse.class, mensaje);

                if (response.getStatus() != 201) {
                        System.out.println("Error. Codigo recibo del Servidor no
valido.");
                } else {
                        System.out.println("OK. Mensaje enviado correctamente.");
                }

        } catch (Exception ex) {
                //System.out.println("Error. " + ex.getMessage());
        }
}
```

Por último y una pieza importante también es la **clase ServicioHost**, esta se encargara de recibir los mensajes de información de los dispositivos que estén conectados a su red de área local. De

esta manera seremos capaces de recibir información tanto a través del puerto serial, como de una red de área local. Para esto, se ha implementado un método el cual se encarga de crear un servicio que está a la escucha en el puerto 5555, de esta manera seremos capaces de ejecutar un nuevo hilo por cada nueva petición de dispositivo hardware que entre a esta API. De esta forma seremos capaces de atender a todos dispositivos que estén conectados.

```
public ServicioHost(int puerto, String dirServicioRest) {

	try {
		sc = new ServerSocket(puerto);
		while(!pararServicio) {
			so = new Socket();
			so = sc.accept();
			hiloCliente = new MainThreadHost(so, dirServicioRest);
			hiloCliente.start();
			so = null;
			}

	} catch(Exception e ) {
		System.out.println("Error: "+e.getMessage());
	}
}
```

Para poder tratar cada dispositivo cuando este se conecte, se creara una **clase MainThreadHost** a la cual se le pasara por parámetro la conexión recibida. Desde esta clase, se invocará al método run(), el cual recibirá los datos enviados por los dispositivos y serán tratados de igual forma que se hacía cuando recibíamos por el puerto serie, por medio de la clase anteriormente mencionada ParserMessage() con el método tratarMensaje().

```
public void run() {
	System.out.println("Hilo iniciado");
	BufferedReader entrada;
	String mensajeRecibido = "";

	try {
		System.out.println("Cliente:" + so.getInetAddress().getHostAddress());
		entrada = new
BufferedReader(newInputStreamReader(so.getInputStream()));
		salida = new DataOutputStream(so.getOutputStream());
		System.out.println("Confirmando conexion al cliente....");
		String mensajeEntrada = entrada.readLine();
		System.out.println(mensajeEntrada);

		tratarMensaje.tratarMensaje(mensajeEntrada);

		salida.writeUTF("200 OK\r\n");

		so.close();

	} catch(Exception ex) {
		System.out.println("Error: " + ex.getMessage());
	}
}
```

Con el objetivo de evitar duplicidades en el código en ciertas clases, se optó por crear una **clase Utiles** de forma que en ella están los métodos más genéricos y que pueden ser empleados por todas las demás clases.

4.2.3 Servidor central REST

Para implementar este servidor central, se creara un proyecto web java con la ayuda del entorno de desarrollo Eclipse. Este proyecto deberá ser configurado de manera que haga uso del SDK de java en la versión 8.

En este proyecto también va a ser necesario hacer uso de otras librerías adicionales por lo que se hará uso de Maven para gestionar las librerías necesarias y de paso para poder generar el fichero de despliegue en un servidor de aplicaciones web como puede ser Tomcat. Para ello primero se creara un proyecto web dinámico y posteriormente se tendrá que convertir dicho proyecto, a un proyecto Maven. Para esto se deberá seleccionar el proyecto y pulsar el botón derecho, al aparecer el menú desplegable pulsaremos sobre “Convert to Maven Proyect” (figura 4.2.3). De esta manera adaptaremos el proyecto y ya podremos gestionar todas las librerías, dependencias y generar el paquete de salida de manera centralizada, dado que toda su configuración está en el fichero pom.xml (figura 4.2.4).

Fig.4.2.3 Conversión proyecto Maven

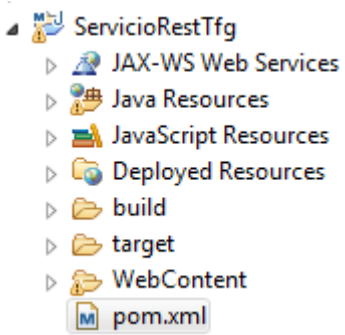


Fig. 4.2.4 Fichero de configuración de Maven

Este fichero pom.xml tendrá añadidas las siguientes dependencias. De esta manera se descargaran todas las librerías necesarias en el proyecto junto a sus dependencias.

```
<dependencies>

	<dependency>
		<groupId>org.json</groupId>
		<artifactId>json</artifactId>
		<version>20151123</version>
	</dependency>

	<dependency>
		<groupId>org.glassfish.jersey.core</groupId>
		<artifactId>jersey-client</artifactId>
```

```
			<version>2.22.2</version>
		</dependency>

		<dependency>
			<groupId>org.glassfish.jersey.containers</groupId>
			<artifactId>jersey-container-servlet</artifactId>
			<version>2.22.2</version>
		</dependency>

		<dependency>
			<groupId>org.mybatis</groupId>
			<artifactId>mybatis</artifactId>
			<version>3.3.1</version>
		</dependency>

		<dependency>
			<groupId>mysql</groupId>
			<artifactId>mysql-connector-java</artifactId>
			<version>5.1.38</version>
		</dependency>

	</dependencies>
```

También será necesario añadir las siguientes líneas al fichero las cuales servirán para compilar y generar el fichero de despliegue .war y al cual se le añadirán todas las dependencias que necesita para funcionar el servidor. En este caso fue necesario forzarle que añadiera los ficheros de configuración de los JavaBean que hace uso la capa de persistencia, así como los ficheros para su configuración dado que todos ellos son ficheros xml.

```
<resources>
	<resource>
		<directory>src/es/upv/beans/config</directory>
		<targetPath>es/upv/beans/config</targetPath>
	</resource>
	<resource>
		<directory>src/es/upv/dao/config</directory>
		<targetPath>es/upv/dao/config</targetPath>
	</resource>
</resources>

<plugins>
	<plugin>
		<artifactId>maven-war-plugin</artifactId>
		<version>2.4</version>
		<configuration>
			<warSourceDirectory>WebContent</warSourceDirectory>
			<failOnMissingWebXml>false</failOnMissingWebXml>
		</configuration>
	</plugin>

	<plugin>
		<artifactId>maven-compiler-plugin</artifactId>
		<version>3.3</version>
		<configuration>
			<source>1.8</source>
			<target>1.8</target>
		</configuration>
	</plugin>

	<plugin>
```

```
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-dependency-plugin</artifactId>
            <version>2.5.1</version>
            <executions>
              <execution>
                   <id>copy-dependencies</id>
                   <phase>package</phase>
                   <goals>
                         <goal>copy-dependencies</goal>
                   </goals>
                   <configuration>
                     <includeScope>runtime</includeScope>
                     <outputDirectory>${project.build.directory}/dependency-
jars/</outputDirectory>
                   </configuration>
              </execution>
            </executions>
      </plugin>
</plugins>
```

Una vez ya está configurado y listo el entorno de desarrollo y tenemos todas las librerías descargas. Ya podemos empezar a implementar la lógica de todo el servidor REST. Debido a la naturaleza del servidor, se optó por separar la lógica de entrada de datos, y la lógica de responder a las peticiones http. De esta manera tendremos dos puntos de entrada al servidor.

Por un lado tenemos el servicio REST el cual es empleado para recibir los mensajes de información enviada por las diferentes API alojadas en las maquinas pasarelas por http vía POST, y almacenar esta información en la base de datos (figura 4.2.5).

Fig. 4.2.5 Flujo seguido por el servicio Rest

Por otro lado tendremos el servicio Rest web, el cual será empleado para recibir peticiones de consulta de información por medio del protocolo http vía GET. Esta información podrá ser utilizada tanto por la aplicación web la cual se encargara de representar toda la información, como por una aplicación externa que requiera de dicha información (figura 4.2.6).

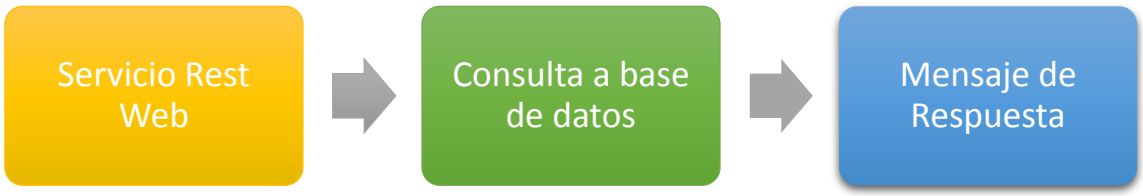


Fig. 4.2.6 Flujo seguido por el servidor Rest Web

Servicio Rest

Este servicio se encarga de recibir los mensajes POST enviados por las diferentes APIS que puedan haber conectadas. Para ello se configuró un servicio Rest con el framework Jersey que

permitirá la recepción de mensajes enviados vía POST o GET. La principal diferencia está en que si recibimos un mensaje vía GET, lo único que obtenemos como respuesta es una simple página web en la que nos informara que tipo de información acepta el servicio.

Para implementar el servicio fue necesario especificar en la **clase ServicioRest** que ruta será la encargada de atender el servicio. Para esto es necesario añadir una anotación "@Path" arriba de la declaración de la clase de la cual queremos que sea un servicio.

```
@Path("/mensajes")
public class ServicioRest {

...

}
```

Las peticiones vía Get son atendidas por un método específico al cual también se le ha tenido que especificar una anotación para ello. Además también es necesario especificar la ruta que va a atender así como el tipo de información que va transferir. Este método devolverá una página en formato html la cual indicara únicamente los tipos de mensaje que son permitidos.

```
@GET
@Path("/add")
@Produces(MediaType.TEXT_HTML)
public String sayHtmlHello() {
	return "<html> " + "<title>" + "Informacion Servicio Rest Tfg" +
	"</title>" + "<body><h1>Informacion Servicio Rest Tfg</h1> + "<p>1 -
	Metodo de entrada por mensaje JSON</p>"                          +
	"<p>2 - Metodo de entrada por mensaje XML</p>"
		+ "</body>" + "</html> ";
}
```

Para recibir los mensajes de información, se han creados dos métodos diferentes donde cada uno es capaz de atender un tipo de mensaje distinto. De esta forma cuando recibimos un mensaje vía POST, la información podría ser un mensaje JSON o bien un mensaje XML. Dado que en el trabajo solo hago uso de mensajes JSON, se ha dejado planteada la idea de que sería posible recibir también mensajes en formato XML de otras fuentes, pero no se ha llegado a implementar dicha funcionalidad y obteniendo un mensaje de respuesta de servicio no disponible.

Para que los métodos sean accesibles, hay que añadir las anotaciones oportunas para especificar el tipo de peticiones que acepta, el tipo de información que admite, en este caso JSON o XML y la ruta en la cual están los recursos, la cual será un conjunto de la ruta anteriormente añadida a la clase junto a la ruta especificada en el método. Ejemplo: /mensajes/add

```
@POST
@Path("/add")
@Consumes(MediaType.APPLICATION_XML)
public Response insertMensajeJson(String msg) {
	System.out.println("Mensaje XML Recibido: " + msg);
	//return Response.status(Response.Status.CREATED).build();
	return Response.status(Response.Status.SERVICE_UNAVAILABLE).build();
}

@POST
@Path("/add")
@Consumes(MediaType.APPLICATION_JSON)
public Response insertMensajeXml(String msg) {
	System.out.println("Mensaje Json Recibido: " + msg);
```

```
        ParserMensajeJson parsearMensaje = new ParserMensajeJson(msg);
        if (parsearMensaje.tratarMensaje()) {
                return Response.status(Response.Status.CREATED).build();
                //return Response.ok("OK",MediaType.TEXT_PLAIN).build();
        } else {
                return Response.status(Response.Status.BAD_REQUEST).build();
        }
}
```

Finalmente como he mencionado anteriormente, únicamente atenderemos peticiones que contengan información en formato JSON. Por tanto una vez hayamos recibido un mensaje de este tipo, este es tratado para averiguar qué tipo de mensaje hemos recibido y así poder realizar la persistencia de los datos en la base de datos. Para ello haremos uso de la **clase ParserMensajeJson.**

Esta clase se encarga una vez recibido el mensaje en formato JSON, de averiguar el tipo de mensaje recibido. Para ello convertimos el mensaje recibido en formato texto plano a un objeto JSON del cual podremos extraer la información de una manera sencilla.

```
JSONObject mensajeJson = new JSONObject(mensaje);
```

A continuación deberemos de comprobar si el mensaje recibo es válido, y si lo es, que tipo de mensaje se ha recibido. Si el tipo de mensaje recibo es correcto, entonces creamos un objeto JavaBean propio para cada tipo con los valores recibidos en el mensaje. En este momento entra en juego la capa de persistencia, por la cual guardaremos la información recibida en la base de datos.

```
if (!mensajeJson.isNull("tipo")) {
        if ("TEM".equals(mensajeJson.getString("tipo"))) {
                MensajePanelTem mensajeTem = new MensajePanelTem();
                mensajeTem.setIdPanel(mensajeJson.getString("idPanel"));
                mensajeTem.setFecha(mensajeJson.getString("fecha"));
                mensajeTem.setHora(mensajeJson.getString("hora"));
                mensajeTem.setValor(mensajeJson.getString("valor"));

                MensajePanelTemDao temperaturaDao = new
        MensajePanelTemDao(MyBatisConnectionFactory.getSqlSessionFactory());
                temperaturaDao.insert(mensajeTem);

                return true;
        } else if ("HUM".equals(mensajeJson.getString("tipo"))) {
                MensajePanelHum mensajeHum = new MensajePanelHum();
                mensajeHum.setIdPanel(mensajeJson.getString("idPanel"));
                mensajeHum.setFecha(mensajeJson.getString("fecha"));
                mensajeHum.setHora(mensajeJson.getString("hora"));
                mensajeHum.setValor(mensajeJson.getString("valor"));

                MensajePanelHumDao humedadDao = new
MensajePanelHumDao(MyBatisConnectionFactory.getSqlSessionFactory());
                humedadDao.insert(mensajeHum);

                return true;
        } else if ("VOL".equals(mensajeJson.getString("tipo"))) {
                MensajePanelVol mensajeVol = new MensajePanelVol();
                mensajeVol.setIdPanel(mensajeJson.getString("idPanel"));
                mensajeVol.setFecha(mensajeJson.getString("fecha"));
                mensajeVol.setHora(mensajeJson.getString("hora"));
                mensajeVol.setValorBat(mensajeJson.getString("valorBat"));
                mensajeVol.setValorPan(mensajeJson.getString("valorPan"));

                MensajePanelVolDao voltajeDao = new MensajePanelVolDao(
MyBatisConnectionFactory.getSqlSessionFactory());
```

```
                voltajeDao.insert(mensajeVol);

                return true;
        } else if ("INT".equals(mensajeJson.getString("tipo"))) {
                MensajePanelInt mensajeInt = new MensajePanelInt();
                mensajeInt.setIdPanel(mensajeJson.getString("idPanel"));
                mensajeInt.setFecha(mensajeJson.getString("fecha"));
                mensajeInt.setHora(mensajeJson.getString("hora"));
                mensajeInt.setValorBat(mensajeJson.getString("valorBat"));
                mensajeInt.setValorPan(mensajeJson.getString("valorPan"));

                MensajePanelIntDao intensidadDao = new MensajePanelIntDao(
MyBatisConnectionFactory.getSqlSessionFactory());
                intensidadDao.insert(mensajeInt);

                return true;
        } else if ("POSPAN".equals(mensajeJson.getString("tipo"))) {
                MensajePanelPos mensajePos = new MensajePanelPos();
                mensajePos.setIdPanel(mensajeJson.getString("idPanel"));
                mensajePos.setFecha(mensajeJson.getString("fecha"));
                mensajePos.setHora(mensajeJson.getString("hora"));
                if (!mensajeJson.isNull("descripcion")) {
                        mensajePos.setDescripcion(mensajeJson.getString("descripcion"));
                }
                mensajePos.setValor(mensajeJson.getString("valor"));
                MensajePanelPosDao posicionDao = new MensajePanelPosDao(
MyBatisConnectionFactory.getSqlSessionFactory());
                posicionDao.insert(mensajePos);

                return true;
        } else if ("LUM".equals(mensajeJson.getString("tipo"))) {
                MensajePanelLum mensajeLum = new MensajePanelLum();
                mensajeLum.setIdPanel(mensajeJson.getString("idPanel"));
                mensajeLum.setFecha(mensajeJson.getString("fecha"));
                mensajeLum.setHora(mensajeJson.getString("hora"));
                mensajeLum.setValorS1(mensajeJson.getString("valorS1"));
                mensajeLum.setValorS2(mensajeJson.getString("valorS2"));
                mensajeLum.setValorS3(mensajeJson.getString("valorS3"));
                mensajeLum.setValorS4(mensajeJson.getString("valorS4"));
                mensajeLum.setValorZonaA(mensajeJson.getString("valorZonaA"));
                mensajeLum.setValorZonaB(mensajeJson.getString("valorZonaB"));
                mensajeLum.setValorUmbralA(mensajeJson.getString("valorUmbralA"));
                mensajeLum.setValorUmbralB(mensajeJson.getString("valorUmbralB"));
                mensajeLum.setValorTotal(mensajeJson.getString("valorTotal"));

                MensajePanelLumDao luminosidadDao = new MensajePanelLumDao
        (MyBatisConnectionFactory.getSqlSessionFactory());
                luminosidadDao.insert(mensajeLum);

                return true;
        } else if ("ERR".equals(mensajeJson.getString("tipo"))) {
                MensajePanelErr mensajeErr = new MensajePanelErr();
                mensajeErr.setIdPanel(mensajeJson.getString("idPanel"));
                mensajeErr.setDescripcion(mensajeJson.getString("descripcion"));
                mensajeErr.setFecha(mensajeJson.getString("fecha"));
                mensajeErr.setHora(mensajeJson.getString("hora"));

                MensajePanelErrDao erroresDAO = new MensajePanelErrDao(
MyBatisConnectionFactory.getSqlSessionFactory());
                erroresDAO.insert(mensajeErr);

                return true;
        } else if ("STA".equals(mensajeJson.getString("tipo"))) {
                MensajePanelStatus mensajeStatus = new MensajePanelStatus();
                mensajeStatus.setIdPanel(mensajeJson.getString("idPanel"));
                mensajeStatus.setDescripcion(mensajeJson.getString("descripcion"));
                mensajeStatus.setFecha(mensajeJson.getString("fecha"));
```

```
            mensajeStatus.setHora(mensajeJson.getString("hora"));

            MensajePanelStatusDao statusDAO = new MensajePanelStatusDao(
        MyBatisConnectionFactory.getSqlSessionFactory());
            statusDAO.insert(mensajeStatus);

            return true;
        }
}
```

Para la capa de persistencia se ha utilizado un Framework llamado myBatis. Este Framework hace uso de un fichero de configuración es/upv/dao/config/config.xml el cual contiene el listado de objetos JavaBeans utilizados, los datos de conexión a la base de datos, así también el listado de ficheros que contiene las querys necesarias para hacer las acciones a la base de datos. A continuación se puede apreciar la estructura de dicho fichero.

```
<configuration>
   <typeAliases>
       <typeAlias alias="MensajePanelStatus" type="es.upv.beans.MensajePanelStatus"/>
       <typeAlias alias="MensajePanel" type="es.upv.beans.MensajePanel"/>
        <typeAlias alias="MensajePanelErr" type="es.upv.beans.MensajePanelErr"/>
        <typeAlias alias="MensajePanelHum" type="es.upv.beans.MensajePanelHum"/>
        <typeAlias alias="MensajePanelInt" type="es.upv.beans.MensajePanelInt"/>
        <typeAlias alias="MensajePanelLum" type="es.upv.beans.MensajePanelLum"/>
        <typeAlias alias="MensajePanelPos" type="es.upv.beans.MensajePanelPos"/>
        <typeAlias alias="MensajePanelTem" type="es.upv.beans.MensajePanelTem"/>
        <typeAlias alias="MensajePanelVol" type="es.upv.beans.MensajePanelVol"/>
        <typeAlias alias="MensajePanelFiltros"
type="es.upv.beans.MensajePanelFiltros"/>
        <typeAlias alias="MensajePanelLumResp"
type="es.upv.beans.MensajePanelLumResp"/>
        <typeAlias alias="MensajePanelTemResp"
type="es.upv.beans.MensajePanelTemResp"/>
        <typeAlias alias="MensajePanelHumResp"
type="es.upv.beans.MensajePanelHumResp"/>
        <typeAlias alias="MensajePanelIntResp"
type="es.upv.beans.MensajePanelIntResp"/>
        <typeAlias alias="MensajePanelVolResp"
type="es.upv.beans.MensajePanelVolResp"/>
    </typeAliases>

    <environments default="development">
        <environment id="development">
          <transactionManager type="JDBC"/>
            <!--  connecting to Local MySql -->
            <dataSource type="POOLED">
                <property name="driver" value="com.mysql.jdbc.Driver"/>
                <property name="url" value="jdbc:mysql://localhost:3306/tfgsolar"/>
                <property name="username" value="root"/>
                <property name="password" value="root"/>
            </dataSource>
       </environment>
    </environments>

    <mappers>
       <mapper resource="es/upv/beans/config/MensajePanelStatus.xml"/>
       <mapper resource="es/upv/beans/config/MensajePanel.xml"/>
       <mapper resource="es/upv/beans/config/MensajePanelErr.xml"/>
       <mapper resource="es/upv/beans/config/MensajePanelHum.xml"/>
       <mapper resource="es/upv/beans/config/MensajePanelInt.xml"/>
       <mapper resource="es/upv/beans/config/MensajePanelLum.xml"/>
       <mapper resource="es/upv/beans/config/MensajePanelPos.xml"/>
       <mapper resource="es/upv/beans/config/MensajePanelTem.xml"/>
       <mapper resource="es/upv/beans/config/MensajePanelVol.xml"/>
       <mapper resource="es/upv/beans/config/MensajePanelLumResp.xml"/>
       <mapper resource="es/upv/beans/config/MensajePanelTemResp.xml"/>
```

```
        <mapper resource="es/upv/beans/config/MensajePanelHumResp.xml"/>
        <mapper resource="es/upv/beans/config/MensajePanelIntResp.xml"/>
        <mapper resource="es/upv/beans/config/MensajePanelVolResp.xml"/>
    </mappers>
</configuration>
```

A continuación es necesario crear una clase para cada tabla existente en el modelo de datos de la base de datos, siempre y cuando se vayan a hacer operaciones con dicha tabla. Estas clases contendrán todos los métodos necesarios para hacer las consultas, modificaciones o inserciones a la base de datos y de esta forma tener más organizado el código. A continuación se exponen un ejemplo de cómo sería un procedimiento de consulta filtrado por fechas e identificador de placa. Es este ejemplo se puede ver como se crea la sesión del objeto encargado de la gestión de la base de datos. Después dado que es necesario pasarle unos parámetros a la consulta SQL, creamos un objeto tipo clave:valor que contendrá dichos parámetros, además tenemos que ser capaces de identificar qué tipo de fecha tenemos para lanzar una query u otra. En este punto lo que hace el propio Framework es obtener la consulta SQL que hemos diseñado anteriormente y que esta ha sido creada en un fichero XML anteriormente.

```
public List<MensajePanelErr> selectByFecha(String fecha, String idplaca) {
        List<MensajePanelErr> listaErrores = null;
        SqlSession session = sqlSessionFactory.openSession();
       try {
                Map<String, Object> params = new HashMap<String, Object>();
                params.put("fecha", fecha);
                params.put("idplaca", idplaca);

                if (fecha.length() == 6) {
                        listaErrores = session.selectList(
"MensajePanelErr.selectByFechaCorta", params);
                } else {
                        listaErrores = session.selectList(
"MensajePanelErr.selectByFechaLarga", params);
                }

         } finally {
             session.close();
         }
        return listaErrores;
}
```

Cada objeto JavaBean (figura 4.2.7) tiene asociado un fichero XML (figura 4.2.8) el cual contiene el mapeo de las columnas de las tablas de la base de datos a los atributos que contiene el propio JavaBean. Además también tiene las sentencias SQL propias de consulta, modificación o creación para esa tabla.

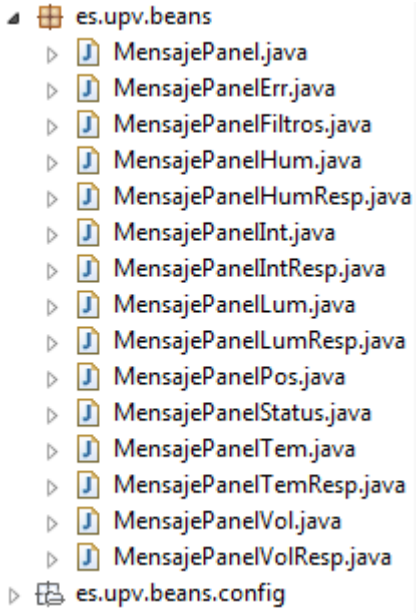


Fig. 4.2.7 Listado de objetos beans

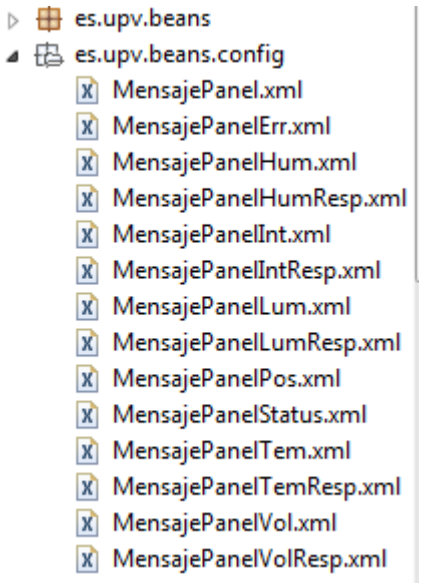


Fig. 4.2.8 Listado de ficheros relacion myBatis

A continuación tenemos el fichero de mapeo para la tabla de errores, en donde podemos apreciar al inicio del fichero, el mapeo de campos de la tabla y atributos del objeto, después están las propias sentencias de consulta, modificación y creación.

```
<mapper namespace="MensajePanelErr">

    <resultMap id="result" type="MensajePanelErr">
        <result property="id" column="id"/>
        <result property="fecha" column="fecha"/>
        <result property="hora" column="hora"/>
        <result property="idPanel" column="idPanel"/>
        <result property="descripcion" column="descripcion"/>
    </resultMap>

    <select id="selectAll" resultMap="result">
        SELECT distinct idpanel, descripcion, fecha, SUBSTR(hora,1,4) as hora FROM
tfgsolar.datos_sensor_err where idpanel = 'PLACA03' order by fecha, hora;
    </select>

    <select id="selectById" parameterType="int" resultMap="result">
        SELECT * FROM datos_sensor_err WHERE id = #{id}
    </select>

    <select id="selectByFechaCorta" parameterType="java.util.Map" resultMap="result">
        SELECT distinct fecha, SUBSTR(hora,1,4) as hora, idPanel, descripcion FROM
tfgsolar.datos_sensor_err terr WHERE SUBSTR(terr.fecha,1,6) = SUBSTR(#{fecha},1,6)
and terr.idPanel = #{idplaca};
    </select>

    <select id="selectByFechaLarga" parameterType="java.util.Map" resultMap="result">
        SELECT distinct fecha, SUBSTR(hora,1,4) as hora, idPanel, descripcion FROM
tfgsolar.datos_sensor_err terr WHERE terr.fecha = #{fecha} and terr.idPanel =
#{idplaca};
    </select>

    <insert id="insert" parameterType="MensajePanelErr" useGeneratedKeys="true"
keyProperty="id">
        INSERT INTO tfgsolar.datos_sensor_err (fecha,hora,idPanel,descripcion) VALUES
(#{fecha}, #{hora}, #{idPanel}, #{descripcion});
    </insert>
```

```
    <update id="update" parameterType="MensajePanelErr">
        UPDATE datos_sensor_err
        SET name = #{name}
        WHERE id = #{id}
    </update>

    <delete id="delete" parameterType="int">
        DELETE from datos_sensor_err WHERE id = #{id}
    </delete>
</mapper>
```

De vuelta al servicio Rest, una vez el mensaje ha sido comprobado, este es insertado a la base de datos por medio de las clases creadas para realizar la persistencia. Para esto se pasa el mensaje por parámetro a la clase creada la cual contendrá la sesión de conexión a la base de datos por medio del método creado para ello.

```
MensajePanelErrDao erroresDAO = new MensajePanelErrDao(
MyBatisConnectionFactory.getSqlSessionFactory());
erroresDAO.insert(mensajeErr);
```

Dicho método se encargara de realizar la persistencia, para ello será se obtendrá la sesión a la base de datos y a continuación se realizará la inserción, para hacer esto se obtendrá la query de inserción del fichero XML.

```
public int insert(MensajePanelErr mensajePanelErr) {

	int id = -1;
	SqlSession session = sqlSessionFactory.openSession();

	try {
		id = session.insert("MensajePanelErr.insert", mensajePanelErr);
	} finally {
		session.commit();
		session.close();
	}
	System.out.println("insert ("+mensajePanelErr+") --> "+
mensajePanelErr.getId());

	return id;
}
```

Una vez realizada la inserción, si todo ha ido correctamente se enviara un código de estado como respuesta indicando que ha ido correctamente la inserción. Si se produjera un error el código de estado devuelto seria de error, así de esta forma podríamos notificar a la API que algo no ha ido bien, pudiéndose tomar acciones alternativas con el fin de no perder la información.

Servicio Rest Web

Este servicio a la diferencia del otro, no procesa mensajes de entrada, sino que es utilizado para responder solicitudes a mensajes de petición. Estos mensajes son recibidos vía GET a una ruta concreta, donde estas rutas definen los diferentes recursos que podemos consultar. Para ellos como en el anterior servicio visto, debemos de especificar una anotación en la clase del servicio.

```
@Path("/web")
public class ServicioRestWeb {

...

}
```

A continuación se implementaran los diferentes métodos por los cuales podremos acceder a consultar la información almacenada en la base de datos. Para realizar estas consultas, se debe de realizar peticiones http vía GET a una ruta en concreto la cual representa un recurso.

A continuación se puede ver el método implementado para realizar la petición de obtener la lista de paneles solares de los cuales tenemos información almacenada. Para dicho método se especifican la anotación para especificar la ruta en la cual se encuentra el recurso, la anotación para especificar el tipo de solicitud que admite, y la anotación para especificar qué tipo de información se obtendrá de él, que en este caso es un mensaje de JSON.

```
@Path("/paneles")
@GET
@Produces(MediaType.APPLICATION_JSON)
public String peticionJsonPaneles(@Context UriInfo uriInfo) {
	System.out.println("Entra peticion paneles.");

	MensajePanelesDao panelesDao = new MensajePanelesDao(
MyBatisConnectionFactory.getSqlSessionFactory());
	List<MensajePanel> lista = null;

	lista = panelesDao.selectAll();

	Iterator nextElement = lista.iterator();
	int cont = 0;
	String mensajePaneles = "";
	while (nextElement.hasNext()) {
		MensajePanel msgPanel = (MensajePanel)nextElement.next();
		if (cont == 0) {
			mensajePaneles += "'" + msgPanel.getIdPanel() + "'";
		} else {
			mensajePaneles += ",'" + msgPanel.getIdPanel() + "'";
		}
		cont = 1;
	}
	System.out.println("MSG: " + mensajePaneles);
	String mensaje = UtilesWeb.obtenerFuncionCallback(uriInfo) + "({";
	mensaje += "paneles:[" + mensajePaneles + "]});";

	return mensaje;
}
```

Desde este método por medio de la capa de persistencia, accedemos a la base de datos de forma parecida a como lo hacíamos en el servicio anterior. Una vez obtenemos el listado de paneles, formamos un mensaje JSON que será devuelto en la respuesta.

Para las peticiones de errores y posiciones es necesario enviar unos parámetros que nos filtren la cantidad de información recibida. Por este motivo lo que se hace es enviar la fecha y el identificador panel en la propia petición GET. Para esto es necesario obtener dichos parámetros en el método de consulta de manera que se pueda invocar el comando SQL con filtros.

```
String filtroFecha = UtilesWeb.obtenerFechaFiltro(uriInfo);

String idplaca = UtilesWeb.obtenerIdPlacaFiltro(uriInfo);
```

Para ello volvemos a preparar la clase encargada de gestionar la conexión a la base de datos y realizamos la consulta por el método oportuno. Todos los métodos de peticiones GET tienen una implementación parecida, es por ello que todo el código mostrado es parecido al que usan el resto.

El código mostrado abajo se encarga de realizar las peticiones de consulta a la base de datos. Estas peticiones son filtradas dado que son pasados por parámetro los valores por los cuales queremos filtrar, en este caso este filtro corresponde a la fecha e identificador de panel. Una vez es ejecutada la consulta, la lista de registros obtenidos serán almacenados en una variable con la cual podremos más adelante utilizar para crear el mensaje de respuesta.

```
MensajePanelPosDao posicionesDao = new MensajePanelPosDao(
MyBatisConnectionFactory.getSqlSessionFactory());
List<MensajePanelPos> lista = posicionesDao.selectByFecha(filtroFecha,
idplaca);
```

Para crear el mensaje de respuesta hacemos uso de la variable anteriormente creada, por la cual recorreremos todos los registros que esta contenga, concatenando los valores de manera que obtengamos un mensaje JSON que pueda ser interpresa de manera simple. Para esta tarea se ha implementado el siguiente código para generar el mensaje, también casi todos los métodos son parecidos en su implementación variando únicamente unos pocos detalles.

```
String mensaje = UtilesWeb.obtenerFuncionCallback(uriInfo) + "([";

Iterator nextElement = lista.iterator();
int cont = 0;
while (nextElement.hasNext()) {
	MensajePanelPos msgPos = (MensajePanelPos)nextElement.next();
	JSONObject mensajeJson = new JSONObject(msgPos);
	if (cont == 0) {
		mensaje += mensajeJson.toString();
	} else {
		mensaje += "," + mensajeJson.toString();
	}
	cont++;
}

mensaje += "]);";
```

Estos métodos tienen una pequeña limitación la cual consiste en que los mensajes de respuesta tienen que tener un prefijo identificativo. Este es necesario por la manera que tiene el Framework AngularJS para atender las peticiones AJAX. Así que si este servicio necesitara ser consultado desde una aplicación externa, se debería de realizar una pequeña modificación en el código.

El listado de métodos implementados para recibir las peticiones serán los siguientes:

```
@Path("/paneles")
@GET
@Produces(MediaType.APPLICATION_JSON)
public String peticionJsonPaneles(@Context UriInfo uriInfo) {...}

@Path("/intensidad")
@GET
@Produces(MediaType.APPLICATION_JSON)
public String peticionJsonIntensidad(@Context UriInfo uriInfo) {...}

@Path("/voltaje")
@GET
@Produces(MediaType.APPLICATION_JSON)
public String peticionJsonVoltaje(@Context UriInfo uriInfo) {...}

@Path("/luminosidad")
@GET
```

```
@Produces(MediaType.APPLICATION_JSON)
public String peticionJsonLuminosidad(@Context UriInfo uriInfo) {...}

@Path("/humedad")
@GET
@Produces(MediaType.APPLICATION_JSON)
public String peticionJsonHumedad(@Context UriInfo uriInfo) {...}

@Path("/temperatura")
@GET
@Produces(MediaType.APPLICATION_JSON)
public String peticionJsonTemperatura(@Context UriInfo uriInfo) {...}

@Path("/posiciones")
@GET
@Produces(MediaType.APPLICATION_JSON)
public String peticionJsonPosiciones(@Context UriInfo uriInfo) {...}

@Path("/errores")
@GET
@Produces(MediaType.APPLICATION_JSON)
public String peticionJsonErrores(@Context UriInfo uriInfo) {...}
```

4.2.4 Portal web

Para la visualización de los datos se implementara una aplicación web la cual nos proporcionara de una manera visual rápida y sencilla los datos recogidos de los distintos paneles solares. Esta aplicación web se implementara en un único fichero el cual será construido por medio de HTML 5 y JavaScript, de manera que este será servido por el propio servidor de aplicaciones sobre el cual se despliegue el paquete de despliegue del servidor principal, dado que este portal va añadido junto al servidor central. De esta forma cualquier petición http que se realice a la dirección “http://IP_host_de_despliegue:8080/ServicioRestTfg/”, recibirá como respuesta el propio portal web de monitorización.

La información que necesita el portal será obtenida por medio de peticiones AJAX, y la respuesta de estas peticiones serán mensajes en formato JSON los cuales serán interpretados y pintados sobre el portal web.

Para realizar toda la lógica JavaScript haremos uso del Framework AngularJS importándolo del CDN de google y así evitando tener que descargarlo. Esto tiene un inconveniente y es que si no se dispone de conexión a internet, no será posible utilizar la aplicación web. Será necesario implementar controladores los cuales manejaran todas las acciones y peticiones necesarias

```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.0/angular.min.js">
</script>
```

También se añadirán las siguientes referencias o librerías necesarias para hacer funcionar la aplicación. Estas corresponden a la librería jquery la cual se utiliza para manipular los objetos DOM de una página web. Aunque AngularJS ya tiene herramientas para modificar los objetos DOM, pero según su propia documentación recomiendan usar como complemento jquery para hacer la manipulación mucho más poderosa.

También se usara bootstrap para la capa de presentación del portal y la librería google charts para pintar los gráficos de los datos.

```
<script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js"> </script>
```

```
<script type="text/javascript"
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"
integrity="sha384-0mSbJDEHialfmuBBQP6A4Qrprq5OVfW37PRR3j5ELqxss1yVqOtnepnHVP9aJ7xS"
crossorigin="anonymous"> </script>

<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js">
</script>
```

Además de las referencias anteriores, se añadirán lo siguientes ficheros JavaScript que contienen los controladores de angularJS que han sido desarrollados y que son necesarios para hacer funcionar la aplicación web.

```
<script type="text/javascript" src="js/jquery-ui.min.js"></script>

<script type="text/javascript" src="js/holder.js"></script>

<script type="text/javascript" src="js/main.js"></script>

<script type="text/javascript" src="js/graficos.js"></script>
```

Como se ha mencionado arriba, se ha hecho uso del Framework AngularJS para implementar la lógica de la aplicación web. Para ello se ha tenido que especificar los controladores que haremos usos en la aplicación en la declaración de AngularJS.

```
angular.module('AppTfgControlPanel', [])
            .controller('menulateral', MenuLateralController)
            .controller('filtrodatos', FiltrosFechaController);
```

Por un lado se ha implementado un controlador para que se haga cargo de las acciones del menú lateral el cual tendrá las secciones que podemos consultar. Por otro lado se creara otro controlador para que se encargue de gestionar el formulario de filtrado de datos de la aplicación. Además también será necesario crear otro el cual nos permitirá controlar el pintar los gráficos en el HTML.

```
angular.module('AppTfgControlPanel')
      .controller('drawGraficos', DrawGraficosController);
```

La aplicación web tendrá una barra superior en la cual se alojaran los filtros (figura 4.2.9) por los que podemos filtrar los datos correspondientes a cada sección. Con este filtro será posible seleccionar el panel solar a través de su identificación, así también el día, mes y años de los datos que queremos consultar.

Fig. 4.2.9 Menú de filtros

Para la implementación del filtro se hizo a través de un formulario con la ayuda de AngularJS al cual cargamos los campos seleccionables a la hora de cargar la página. En este caso es AngularJS el que hace la "magia" por nosotros e interpreta las variables de su controlador asociado para pintar el valor de dichas variables en el código HTML.

```
<form class="navbar-form navbar-right bloqueFiltros" ng-controller="filtrodatos">
       <div class="form-group" >
              <label class="textos" for="dateFilter">Filtrar por:</label>
              <select class="form-control input-sm" ng-model="panelSelected"
name="panel">
```

```
                    <option value="{{panel}}" ng-repeat="panel in listadoPaneles"
ng-selected="{{panel == panelActual}}">{{panel}}</option>
                </select>
                <select class="form-control input-sm" ng-model="diaSelected" name="dia"
ng-hide="ocultarSelectDias">
                    <option value="{{dia}}" ng-repeat="dia in listadoDias" ng-
selected="{{dia == diaActual}}">{{dia}}</option>
                </select>
                <select class="form-control input-sm" ng-model="mesSelected"
name="mes">
                    <option value="{{mes}}" ng-repeat="mes in listadoMeses" ng-
selected="{{mes == mesActual}}">{{mes}}</option>
                </select>
                <select class="form-control input-sm" ng-model="anyoSelected"
name="anyo">
                    <option value="{{anyo}}" ng-repeat="anyo in listadoAnyos" ng-
selected="{{anyo == anyoActual}}">{{anyo}}</option>
                </select>
        </div>
        <button type="submit" class="btn btn-default btn-sm" ng-click="doFilter()">
Filtrar</button>
</form>
```

El controlador encargado de cargar los datos de los campos seleccionables del filtro se llama *FiltrosFechaController* y se establece por medio de la etiqueta *ng-controller* a la cual se le indica el nombre del controlador, este se implementó de manera que obtiene la fecha actual como referencia, pero además añade todos los posibles valores fijados para que pueda ser seleccionado cualquier rango de fechas. En principio este rango esta puesto de manera fija, pero una posible modificación a tener en cuenta seria que solo se añadieran en los campos seleccionables las fechas de las cuales dispongamos de información. También se obtiene el listado de paneles que hay almacenados en la base de datos en donde más adelante se explicara de qué manera se obtiene la información.

```
managerPeticiones("LISTPAN", $scope, $http);

$scope.filtroSeleccionado = "LUM";
$scope.modoQuerySelect = "TOTAL";

$scope.ocultarSelectDias = true;

$scope.listadoDias = ["All", "01", "02", "03", "04", "05", "06", "07", "08", "09",
"10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21", "22", "23",
"24", "25", "26", "27", "28", "29", "30", "31"];
$scope.listadoMeses = ["Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio",
"Julio", "Agosto", "Septiembre", "Octubre", "Noviembre", "Diciembre"];
$scope.listadoAnyos = ['2014', '2015', '2016', '2017', '2018', '2019', '2020'];

$scope.loadFilter = function() {
        $scope.fecha = new Date();
        if ((($scope.fecha.getDate().toString()).length) == 1) {
                $scope.diaActual = "0" + ($scope.fecha.getDate()).toString();
        } else {
                $scope.diaActual = ($scope.fecha.getDate()).toString();
        }
        $scope.diaSelected = $scope.diaActual;
        $scope.mesActual = $scope.listadoMeses[$scope.fecha.getMonth()];
        $scope.mesSelected = $scope.mesActual;
        $scope.anyoActual = $scope.fecha.getFullYear().toString();
        $scope.anyoSelected = $scope.anyoActual;
        $scope.panelActual = "PLACA03";
        $scope.panelSelected = $scope.panelActual;
}

$scope.loadFilter();
```

Para organizar el contenido de la aplicación, se decidió tener un panel a la izquierda (figura 4.2.10) el cual nos ofrezca todas las posibles opciones de consulta. Para ello se dividió la información en las siguientes secciones:

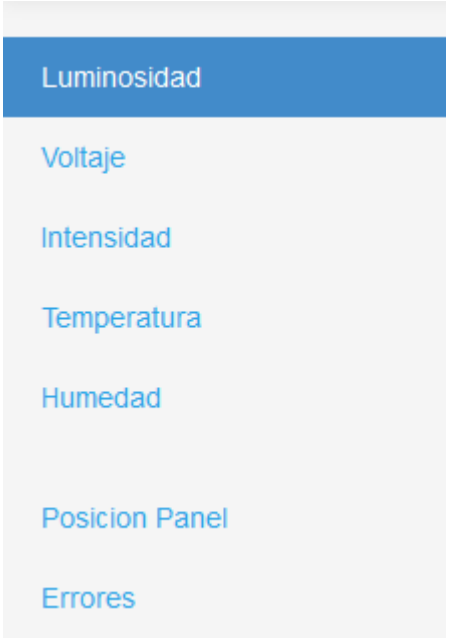


Fig. 4.2.10 Menú lateral con secciones

En esta implementación, realmente cada sección muestra una información bastante vaga, pero la cual nos sirve para hacernos una idea de que sería capaz de hacer todo el conjunto si se llevara a cabo el implementar esta solución a un entorno real.

Para la implementación de este panel lateral, se desarrolló otro controlador llamado *MenuLateralController* el cual se encarga de mostrar u ocultar las regiones de contenido correspondientes a cada sección. De este modo se consigue tener una única página que es capaz de visualizar diversa información.

```
<div class="row" ng-controller="menulateral">
        <div class="col-sm-3 col-md-2 sidebar" >
          <ul class="nav nav-sidebar">
            <li ng-class="linkActivoLum"><a href="#" ng-click="ShowHideLum()">
Luminosidad</a></li>
            <li ng-class="linkActivoVol"><a href="#" ng-click="ShowHideVol()">
Voltaje</a></li>
            <li ng-class="linkActivoInt"><a href="#" ng-click="ShowHideInt()">
Intensidad</a></li>
            <li ng-class="linkActivoTem"><a href="#" ng-click="ShowHideTem()">
Temperatura</a></li>
            <li ng-class="linkActivoHum"><a href="#" ng-click="ShowHideHum()">
Humedad</a></li>
          </ul>
          <ul class="nav nav-sidebar">
            <li ng-class="linkActivoPos"><a href="" ng-click="ShowHidePos()">Posicion
Panel</a></li>
            <li ng-class="linkActivoErr"><a href="" ng-click="ShowHideErr()">
Errores</a></li>
          </ul>
        </div>
...
</div>
```

Para realizar la acción de mostrar u ocultar la región con la información, se hizo uso de los eventos que permite AngularJS. De este modo se añadió a los elementos de la lista la etiqueta *ng-click* la cual nos permite decirle que función del controlador queremos ejecutar. De esta forma es sencilla cambiar las propiedades de visibilidad de las regiones de información, además se creó un método para ayudar en estas tareas `controlarMenuLateral` el cual nos ayuda a indicar si se oculta o no la región y si esta está activa.

```
$scope.ShowHideTem = function () {
	controlarMenuLateral($scope, true, true, true, false, true, true, true, "",
"", "", "active", "", "", "");
	$scope.reloadFilter();
	$scope.ocultarSelectDias(true);
	$scope.CargarDatosFiltros('TEM');
}

function controlarMenuLateral(varglobal, err, int, vol, tem, hum, pos, lum, lerr,
lint, lvol, ltem, lhum, lpos, llum) {
	 varglobal.IsHiddenErr = err;
	 varglobal.IsHiddenInt = int;
	 varglobal.IsHiddenVol = vol;
	 varglobal.IsHiddenTem = tem;
	 varglobal.IsHiddenHum = hum;
	 varglobal.IsHiddenPos = pos;
	 varglobal.IsHiddenLum = lum;

	 varglobal.linkActivoTem = ltem;
	 varglobal.linkActivoHum = lhum;
	 varglobal.linkActivoLum = llum;
	 varglobal.linkActivoInt = lint;
	 varglobal.linkActivoVol = lvol;
	 varglobal.linkActivoPos = lpos;
	 varglobal.linkActivoErr = lerr;
 };
```

Para la obtención de los datos haremos uso de una función llamada `managerPeticiones`, a la cual se le indicara que tipo de dato se requiere y además se le pasa las variables del contexto del controlador. Esta función es importante porque es la que va a tener todas las rutas a todos los recursos que se deben de consultar para obtener los datos necesarios. Para ello se tiene una función para cada tipo de recurso disponible, de modo que sería posible tener en esta misma función una consulta a un recurso por medio de una misma función pero con distinta ruta de acceso al recurso.

```
function managerPeticiones(tipoPeticion, contexto, conexion, filtros, global) {

	if (tipoPeticion == "ERR") {
		getDataFromRestErr("/ServicioRestTfg/rest/web/errores", contexto,
conexion, filtros, global);
	} else if (tipoPeticion == "POS") {
		getDataFromRestPos("/ServicioRestTfg/rest/web/posiciones", contexto,
conexion, filtros, global);
	} else if (tipoPeticion == "TEM") {
		getDataFromRestTemTotal("/ServicioRestTfg/rest/web/temperatura",
contexto, conexion, filtros);
	} else if (tipoPeticion == "HUM") {
		getDataFromRestHumTotal("/ServicioRestTfg/rest/web/humedad", contexto,
conexion, filtros);
	} else if (tipoPeticion == "LUM") {
		getDataFromRestLumTotal("/ServicioRestTfg/rest/web/luminosidad",
contexto, conexion, filtros);
	} else if (tipoPeticion == "INT") {
		getDataFromRestIntTotal("/ServicioRestTfg/rest/web/intensidad",
contexto, conexion, filtros);
	} else if (tipoPeticion == "VOL") {
```

```
		getDataFromRestVolTotal("/ServicioRestTfg/rest/web/voltaje", contexto,
conexion, filtros);
	} else if (tipoPeticion == "LISTPAN") {
		getDataFromRestListPaneles("/ServicioRestTfg/rest/web/paneles",
contexto, conexion);
	}

 }
```

Como aclaración a lo comentado en el párrafo anterior, tenemos estas funciones las cuales se encargan de hacer las peticiones AJAX al servicio REST del servidor central. Dado que el propio Framework AngularJS me permite obtener recursos JSON de forma directa gracias a la función *jsonp,* puedo trabajar directamente con los mensajes recibidos sin necesidad de tratarlos, pudiéndolos pintar en la página web de forma inmediata.

```
function getDataFromRestTemTotal(url, global, conexion, filtros) {
	var c4 = conexion.jsonp(url + "?callback=JSON_CALLBACK" + filtros);
		c4.success(function(response4) {
			global.datosTemTotal.addRows(response4);
			global.graficoTemTotal.draw(global.datosTemTotal,
global.optionsTem);
		});

};
```

Para la representación gráfica de los datos, se ha optado por hacer uso de la librería que ofrece google. Charts nos ofrece un Framework completo con el cual podemos crear una variedad elevada de gráficos de manera sencilla y rápida. Esto fue implementado en otro controlador llamado *DrawGraficosController* y el cual contendrá todas las funciones encargadas de representar los datos de las diferentes secciones en un formato gráfico.

Estas funciones tendrán todas ellas las configuraciones necesarias para el grafico que se quiere pintar.

```
$scope.graficosHum = function(modo, panel, fecha) {

		if (fecha === undefined) {
			$scope.filtroFecha = null;
		} else {
			$scope.filtroFecha = fecha;
		}
		$scope.filtroModo = modo;
		$scope.filtroPanel = panel;
		$scope.optionsHum = {
				width: 850,
				height: 350,
				pointSize: 5,
				hAxis: { ticks: [1,3,6,9,12,15,18,21,24,27,30], title:
"Dias"}
			};

		if (modo == "TOTAL") {
			google.charts.setOnLoadCallback(
				function() {
					DrawGraficosHumTotal($scope, $http);
					managerPeticiones("HUM", $scope, $http,
$scope.filtros);
				}
			);
		}
	};
```

Posteriormente se invocara a una función particular la cual establecerá los nombres de las columnas del datasets recibido, así como el contenedor HTML donde será pintado. Esta función es posible que fuera una cosa a revisar y a modificar en una mejora dado que podríamos haber implementado su funcionalidad en la función superior evitando hacer otra llamada más.

```
function DrawGraficosHumTotal (variable, conexion) {
	variable.datosHumTotal = new google.visualization.DataTable();
	variable.datosHumTotal.addColumn('number', 'Dias');
	variable.datosHumTotal.addColumn('number', 'Media');
	variable.datosHumTotal.addColumn('number', 'Max');
	variable.datosHumTotal.addColumn('number', 'Min');

	if (variable.filtroFecha == null) {
		variable.filtros = generarFiltroFechaActual(false) +
generarFiltroModo(variable.filtroModo) + generarFiltroIdPanel(variable.filtroPanel);
	} else {
		variable.filtros = generarFiltroFecha(variable.filtroFecha) +
generarFiltroModo(variable.filtroModo) + generarFiltroIdPanel(variable.filtroPanel);
	}

	variable.graficoHumTotal = new
google.visualization.LineChart(document.getElementById('panel_hum'));

}
```

Para finalizar el proceso de representación de los gráficos, debemos obtener los datos del servidor, para ello se realiza una petición REST vía GET la cual se realizara mediante el método visto anteriormente llamado *managerPeticiones.* Cuando esta función invoque a la que realice la petición real, esta recibirá la información por medio de un mensaje JSON el cual será añadido al dataset del objeto del gráfico, y ya podremos invocar la orden de pintar el grafico en el contenedor HTML.

```
function getDataFromRestHumTotal(url, global, conexion, filtros) {

	var c5 = conexion.jsonp(url + "?callback=JSON_CALLBACK" + filtros);
		c5.success(function(response5) {
			global.datosHumTotal.addRows(response5);
			global.graficoHumTotal.draw(global.datosHumTotal,
global.optionsHum);
		});

};
```

A continuación se presentara una captura de cada sección implementada con el fin de que pueda apreciarse el resultado final de la aplicación web.

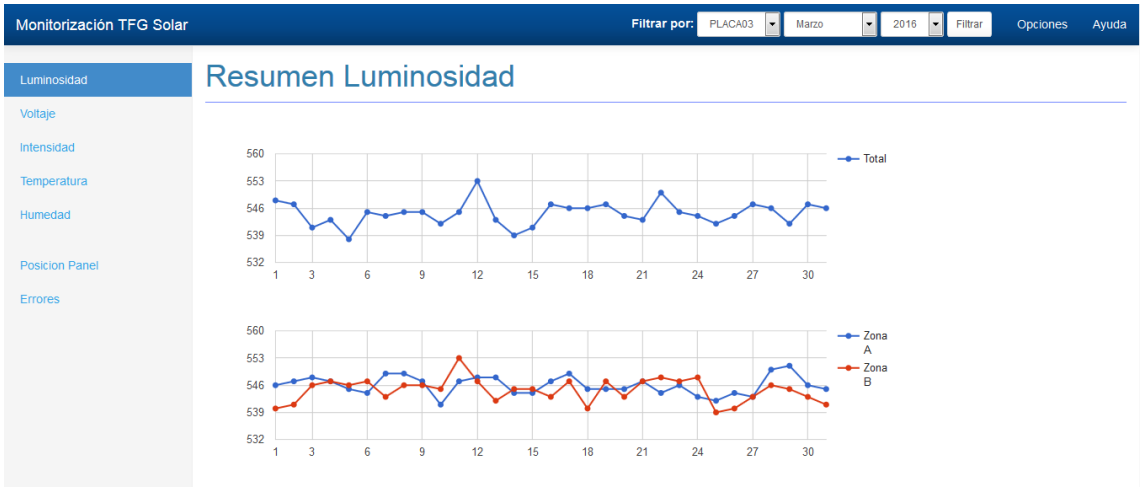


Fig. 4.2.11 Sección Luminosidad

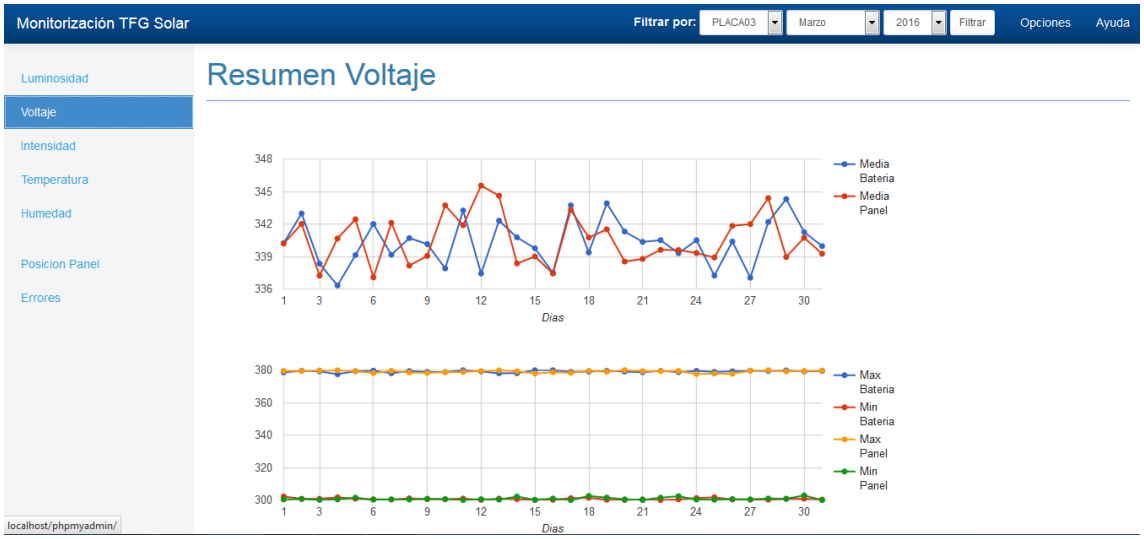


Fig. 4.2.12 Sección Voltaje

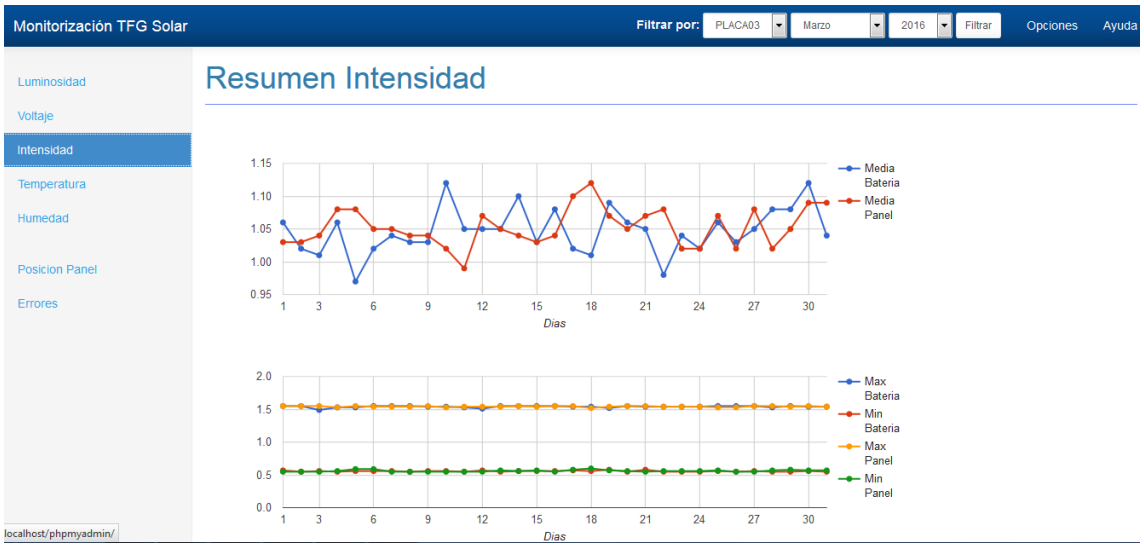


Fig. 4.2.13 Sección Intensidad

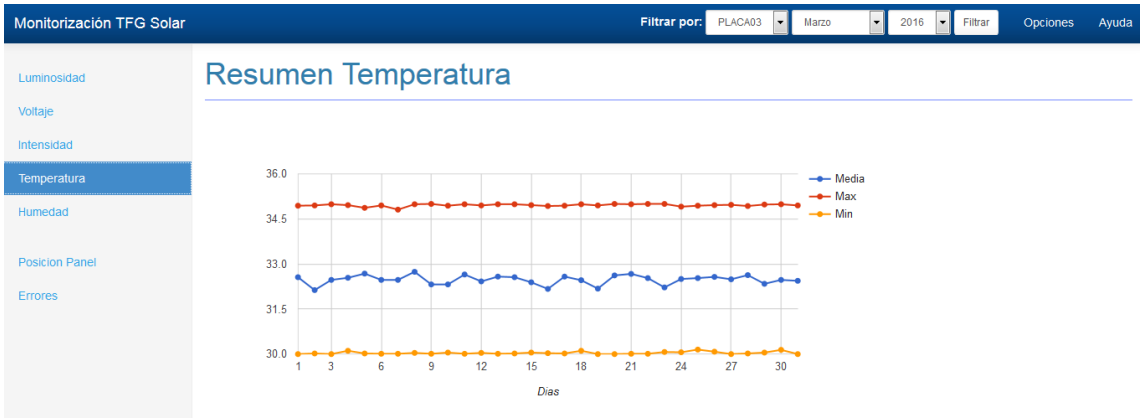


Fig. 4.2.14 Sección Temperatura

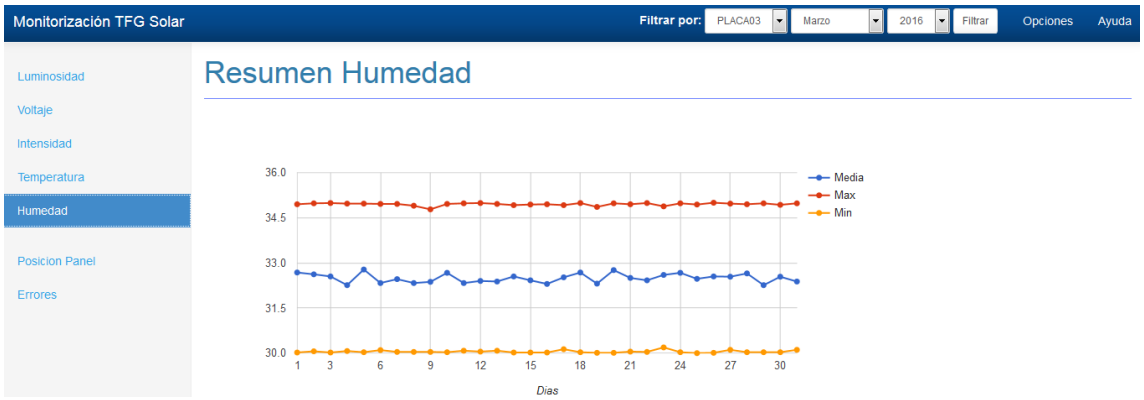


Fig. 4.2.15 Sección Humedad

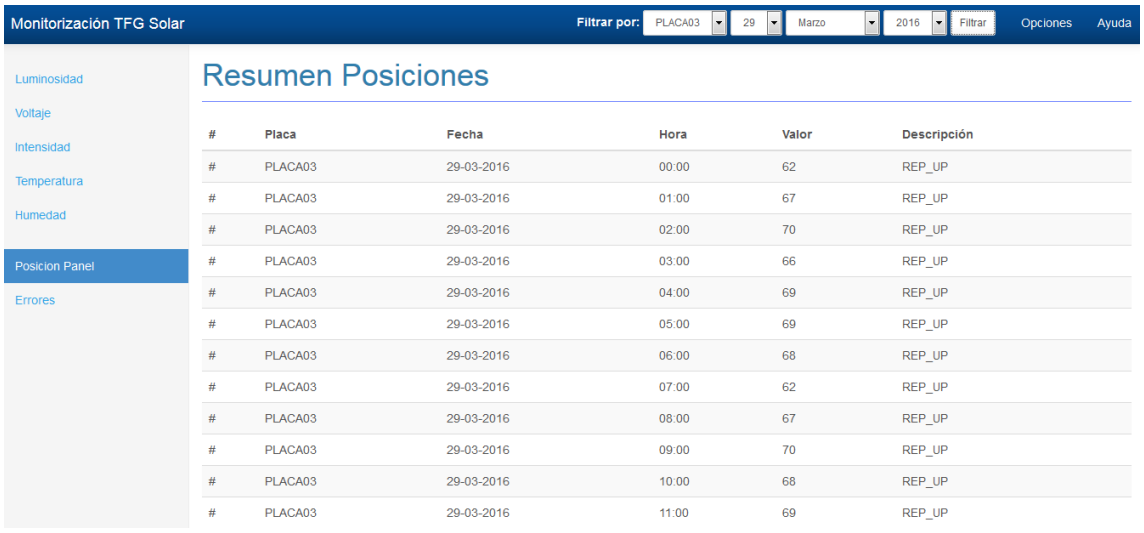


Fig. 4.2.16 Sección posiciones

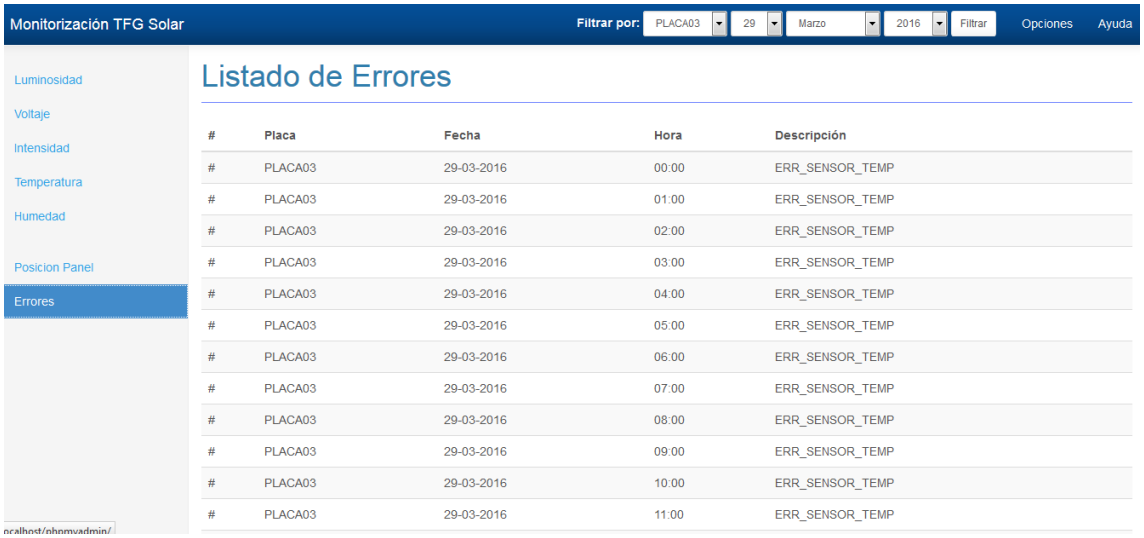


Fig. 4.2.17 Sección Errores

4.2.5 Creación del modelo de base de datos

Para almacenar la información en la base de datos, es necesario crear los contenedores para albergar la información. Estos contenedores son las tablas en las cuales se insertaran los datos en forma de registro por medio de columnas, en donde cada columna albergara un tipo de información.

Para ello primero es necesario crear una base de datos (figura 4.2.18) en la cual podamos almacenar la información. Para ello tenemos que ir al gesto de la base de datos de MySQL y crearla.

Fig. 4.2.18 Asistente creación base de datos

Una vez tenemos la base de datos, creamos unas consultas de borrado para asegurarnos que no existen dichas tablas o también para eliminarlas de forma rápida en cualquier momento.

- DROP TABLE tfgsolar.DATOS_SENSOR_LUM;
- DROP TABLE tfgsolar.DATOS_SENSOR_TEM;
- DROP TABLE tfgsolar.DATOS_SENSOR_HUM;
- DROP TABLE tfgsolar.DATOS_SENSOR_VOL;
- DROP TABLE tfgsolar.DATOS_SENSOR_INT;
- DROP TABLE tfgsolar.DATOS_SENSOR_POS;
- DROP TABLE tfgsolar.DATOS_SENSOR_ERR;
- DROP TABLE tfgsolar.DATOS_SENSOR_STATUS;

Una vez se ha comprobado que no existen las tablas ya creadas, lanzamos las siguientes instrucciones las cuales crearan todas las tablas necesarias para almacenar toda la información recibida en el servidor Rest.

```
CREATE TABLE tfgsolar.DATOS_SENSOR_LUM (
        id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
        fecha VARCHAR(8) NOT NULL,
        hora VARCHAR(6) NOT NULL,
        idPanel VARCHAR(30) NOT NULL,
        luminosidad1 VARCHAR(6),
        luminosidad2 VARCHAR(6),
        luminosidad3 VARCHAR(6),
        luminosidad4 VARCHAR(6),
        luminosidadZona1 VARCHAR(6),
        luminosidadZona2 VARCHAR(6),
        umbralLuminosidadZona1 VARCHAR(6),
        umbralLuminosidadZona2 VARCHAR(6),
        luminosidadTotal VARCHAR(6)
);

CREATE TABLE tfgsolar.DATOS_SENSOR_TEM (
        id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
        fecha VARCHAR(8) NOT NULL,
        hora VARCHAR(6) NOT NULL,
        idPanel VARCHAR(30) NOT NULL,
        temperatura VARCHAR(6)
);

CREATE TABLE tfgsolar.DATOS_SENSOR_HUM (
        id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
        fecha VARCHAR(8) NOT NULL,
        hora VARCHAR(6) NOT NULL,
        idPanel VARCHAR(30) NOT NULL,
        humedad VARCHAR(6)
);

CREATE TABLE tfgsolar.DATOS_SENSOR_VOL (
        id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
        fecha VARCHAR(8) NOT NULL,
        hora VARCHAR(6) NOT NULL,
        idPanel VARCHAR(30) NOT NULL,
        voltajePlaca VARCHAR(6),
        voltajeBateria VARCHAR(6)
);

CREATE TABLE tfgsolar.DATOS_SENSOR_INT (
        id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
        fecha VARCHAR(8) NOT NULL,
        hora VARCHAR(6) NOT NULL,
        idPanel VARCHAR(30) NOT NULL,
        intensidadPlaca VARCHAR(6),
        intensidadBateria VARCHAR(6)
);

CREATE TABLE tfgsolar.DATOS_SENSOR_POS (
        id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
        fecha VARCHAR(8) NOT NULL,
        hora VARCHAR(6) NOT NULL,
```

```
        idPanel VARCHAR(30) NOT NULL,
        posicion VARCHAR(6),
        descripcion VARCHAR(50)
);

CREATE TABLE tfgsolar.DATOS_SENSOR_ERR (
        id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
        fecha VARCHAR(8) NOT NULL,
        hora VARCHAR(6) NOT NULL,
        idPanel VARCHAR(30) NOT NULL,
        descripcion VARCHAR(50)
);

CREATE TABLE tfgsolar.DATOS_SENSOR_STATUS (
        id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
        fecha VARCHAR(8) NOT NULL,
        hora VARCHAR(6) NOT NULL,
        idPanel VARCHAR(30) NOT NULL,
        descripcion VARCHAR(50)
);
```

Una vez tenemos las tablas creadas en la base de datos, estas estarán vacías inicialmente por lo que es imposible obtener datos para representarlos en el portal web. Por ello se han creado varios procedimientos por los cuales podemos generar datos de prueba para cada tabla. A continuación se puede ver un procedimiento el cual corresponde a la tabla de humedad, pero realmente el resto de procedimiento es relativamente parecido existiendo pequeñas diferencias.

```
drop procedure if exists tfgsolar.GenerarDatosTestHum;

CREATE PROCEDURE tfgsolar.GenerarDatosTestHum(IN anyo int(4)) MODIFIES SQL DATA
begin
 declare FechaInicio VARCHAR(10);
 declare HoraInicio VARCHAR(10);
 declare mes int;
 declare dia int;
 declare hora int;
 declare UltimoDia int;
 set mes := 1;
 TRUNCATE TABLE tfgsolar.datos_sensor_hum;
 while mes < 12 do
  IF length(mes) < 2 then
   set FechaInicio := concat(anyo, '0', mes, '01');
  else
   set FechaInicio := concat(anyo, mes, '01');
  end IF;
  set dia := 1;
  set UltimoDia := CAST(SUBSTR(LAST_DAY(FechaInicio),9,10) AS UNSIGNED);
  while dia <= UltimoDia do
   set hora := 0;
   while hora < 24 do
    IF length(hora) < 2 then
     set HoraInicio := concat('0',hora, '0000');
    else
```

```
        set HoraInicio := concat(hora, '0000');
      end IF;
      INSERT INTO tfgsolar.datos_sensor_hum(fecha, hora, idPanel, humedad) VALUES
(FechaInicio,HoraInicio,'PLACA03',round(30 + (rand() * 5),2));
      IF length(hora) < 2 then
        set HoraInicio := concat('0',hora, '1500');
      else
        set HoraInicio := concat(hora, '1500');
      end IF;
      INSERT INTO tfgsolar.datos_sensor_hum(fecha, hora, idPanel, humedad) VALUES
(FechaInicio,HoraInicio,'PLACA03',round(30 + (rand() * 5),2));
      IF length(hora) < 2 then
        set HoraInicio := concat('0',hora, '3000');
      else
        set HoraInicio := concat(hora, '3000');
      end IF;
      INSERT INTO tfgsolar.datos_sensor_hum(fecha, hora, idPanel, humedad) VALUES
(FechaInicio,HoraInicio,'PLACA03',round(30 + (rand() * 5),2));
      IF length(hora) < 2 then
        set HoraInicio := concat('0',hora, '4500');
      else
        set HoraInicio := concat(hora, '4500');
      end IF;
      INSERT INTO tfgsolar.datos_sensor_hum(fecha, hora, idPanel, humedad) VALUES
(FechaInicio,HoraInicio,'PLACA03',round(30 + (rand() * 5),2));
      set hora := hora + 1;
    end while;
    set FechaInicio := REPLACE(DATE_ADD(FechaInicio,INTERVAL 1 day),'-','');
    set dia := dia + 1;
  end while;
  set mes := mes + 1;
 end while;
end
CALL tfgsolar.GenerarDatosTest(2016)
```

En la siguiente figura se detallará el resumen de los objetos creados y que deben de existir en la base de datos.

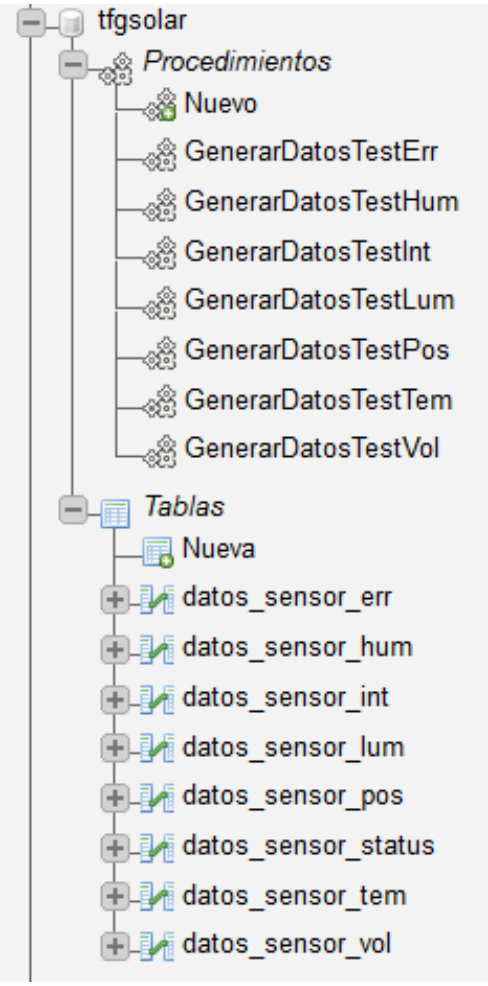


Fig. 4.2.19 Objetos credos en la base de datos

5. Análisis de resultados

Una vez terminada toda la etapa de implementación del prototipo se han realizado unas pruebas con las cuales obtener valores y comprobar si realmente se obtiene una mejora con la modificación realizada al panel solar respecto al típico panel solar ubicado de manera fija. Estas pruebas van a consistir en tomar dos muestras de días consecutivos de valores para cada tipo de instalación de panel solar, de manera que para estas pruebas el panel va a estar siempre ubicado en el mismo lugar intentado igualar las condiciones para todas las pruebas. La muestras serán tomadas cada 1 minuto y para obtener el valor de muestra se realizara la media para cada hora, obteniendo una valor medio por hora.

El panel solar será ubicado de manera que apuntará en dirección al este, estando en estado de reposo, así de esta forma conforme el día vaya avanzando, el panel solar seguirá al sol cuando este se desplace hacia el oeste como puede verse en la figura inferior.

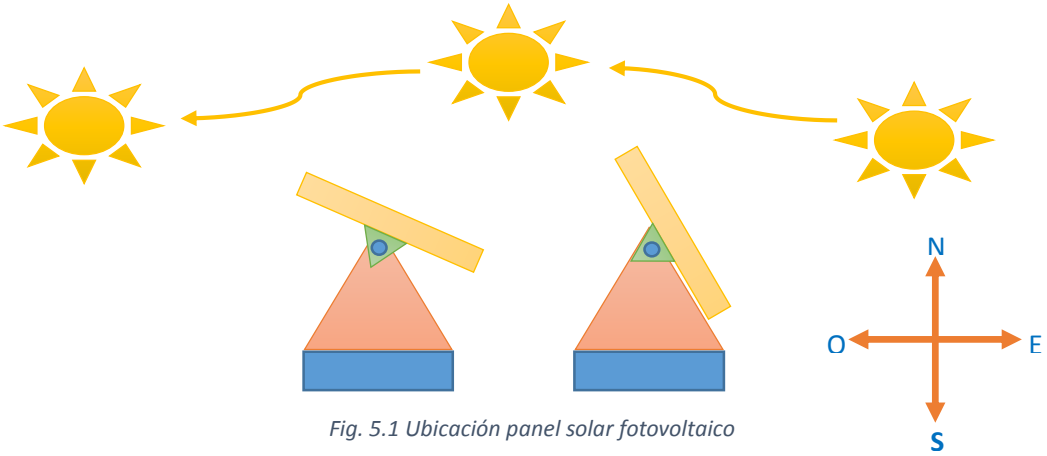
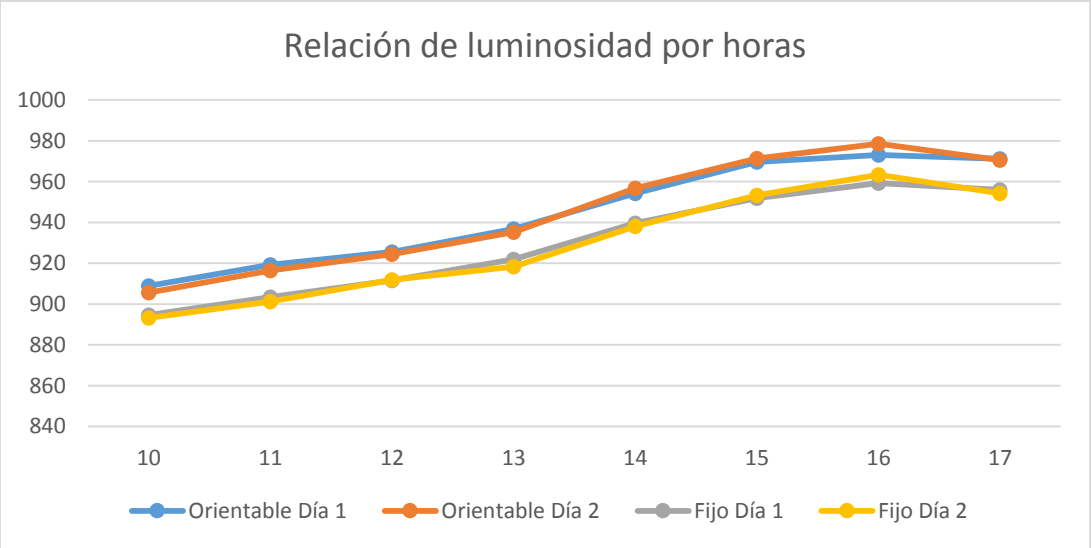


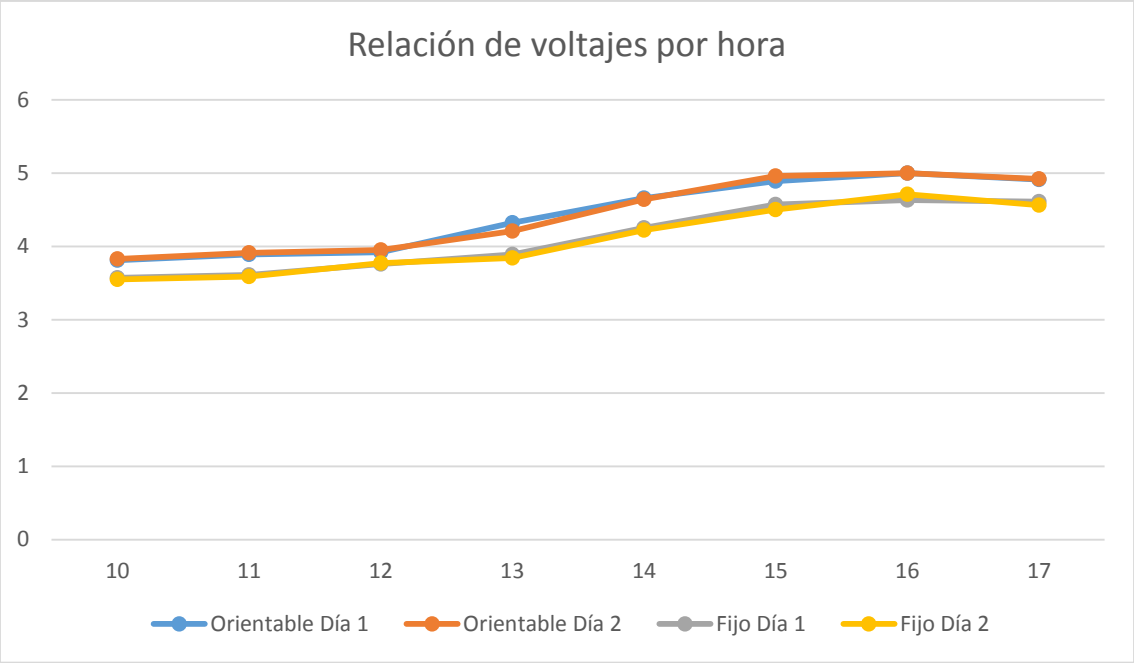
Fig. 5.1 Ubicación panel solar fotovoltaico

A continuación se van a exponer los datos obtenidos a través del dispositivo de monitorización y se va a proceder a comparar dichos resultados.

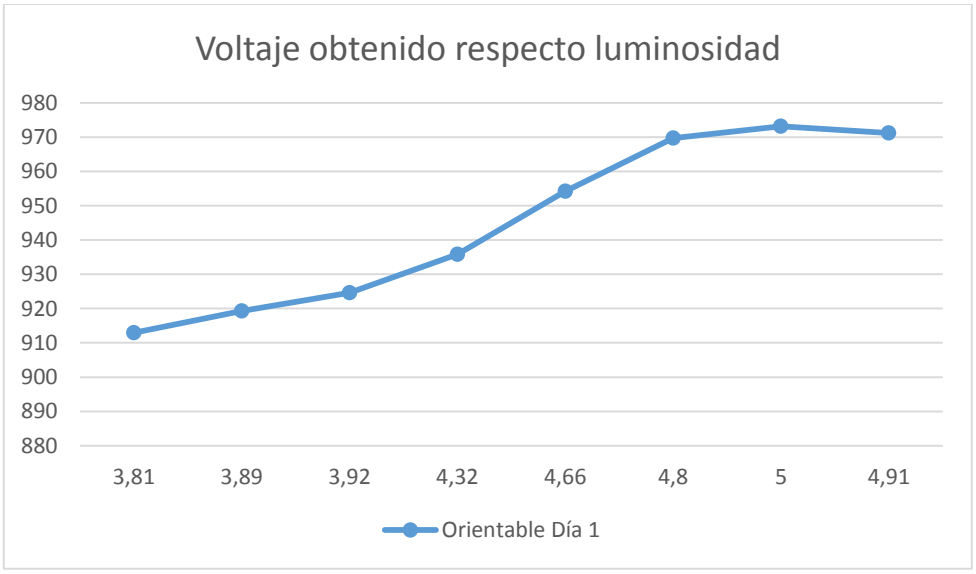
Por medio de los datos obtenidos se ha podido comprobar que efectivamente se consigue una pequeña mejoria en la captacion de luminosidad por parte del panel solar fotovoltaico. Como se puede apreciar en el grafico inferior los valores obtenidos por el panel solar orientable para los dos días son un poco mas elevados de manera que ha recibido mas luminosidad que el panel solar estando en modo fijo en donde sus valores se situan por debajo.

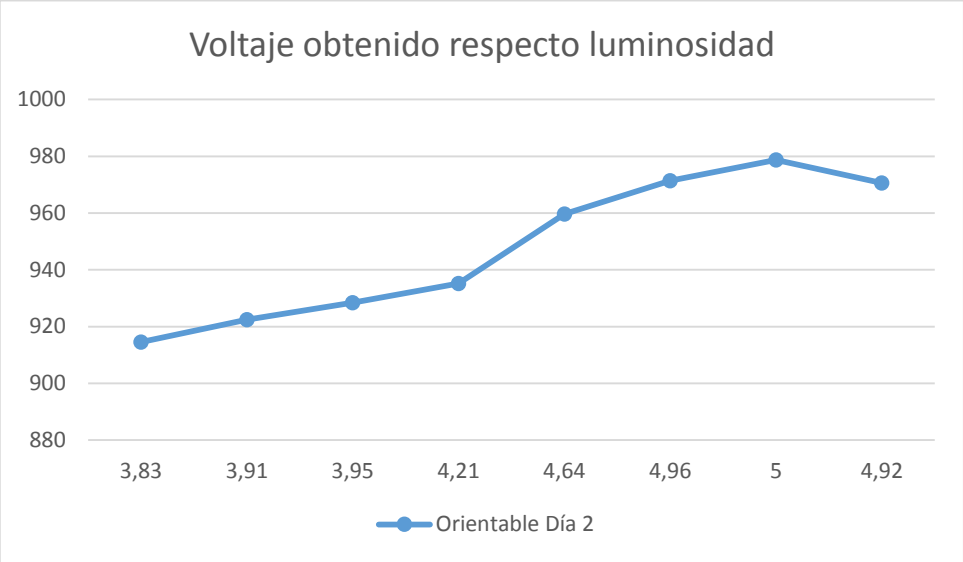


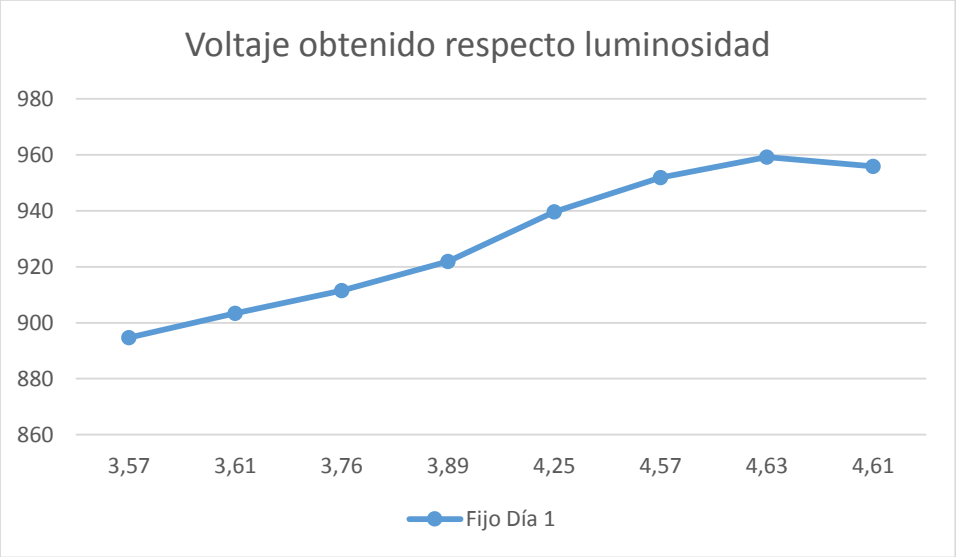
También se ha podido comprobar que gracias a esta mejora añadida al panel solar, este incrementa su voltaje de salida a la batería. Consiguiendo de esta manera que se pueda empezar a cargar las baterías mucho antes, aprovechando el máximo sol del que se dispone. Como se puede ver en el grafico inferior, se puede apreciar el contraste entre emplear un panel solar orientable o un panel solar fijo. Cuando se hace uso del panel solar orientable, es capaz de ofrecer el 100% de la salida energética de lo máximo que es capaz de ofrecer este modelo de panel solar. Esto se traduce a que es capaz de ofrecernos los 18 voltios que ofrece el panel y una intensidad de 0,27 A, dando una generación de 5 vatios/hora. De la otra forma no se llega a obtener la máxima generación quedándose a valores cercanos.

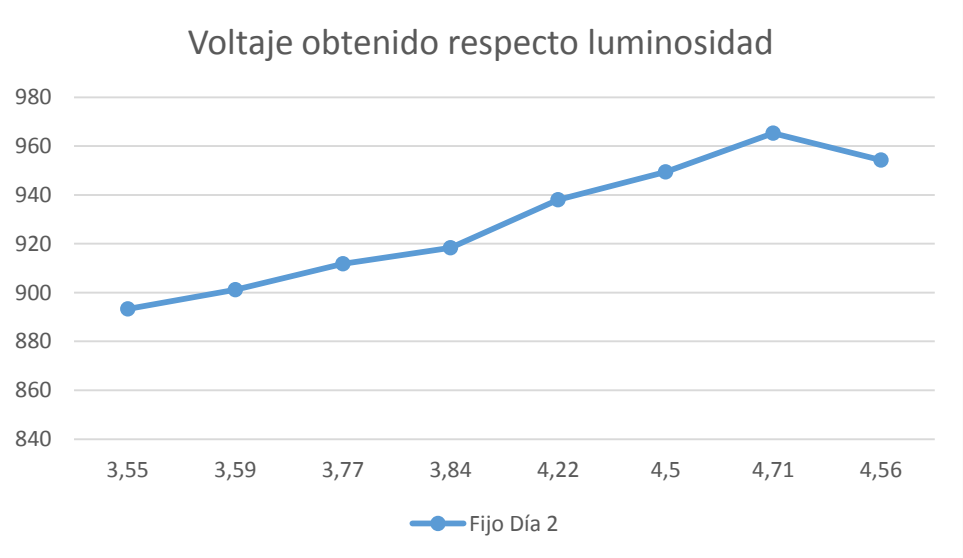


A continuación se puede apreciar por medio de las siguientes graficas como afecta la luminosidad recibida en el panel solar en los dos diferentes modos de funcionamiento a la generación de energía obtenida por parte de este. Observándose como a mayor luminosidad recibida en el panel solar, mayor es el voltaje generado y proporcionalmente mayor será la intensidad generada por este.









A partir de los valores de luminosidad obtenidos podemos calcular la eficiencia ganada empleando la mejora añadida al panel solar. Para ello se ha calculado la media de las muestras obtenidas durante los dos días y de esta forma obtenemos que para el panel ubicado de manera

fija, obtenemos un valor de 929,48 de luminosidad y para el panel ubicado de forma orientable al sol obtenemos un valor medio de 944,86. De esta manera podemos calcular el rendimiento obtenido de la mejora obteniendo así una ganancia de 1,66% respecto al panel ubicado de manera fija.

$$\left(\frac{\mathbf{944,86}}{\mathbf{929,48}} - \mathbf{1}\right) * \mathbf{100} = \mathbf{1,66}\%$$

Si estos mismos cálculos los aplicamos también para ver el rendimiento obtenido en el voltaje generado por el panel solar, podemos apreciar que obtenemos una mejora de un 7,91% de manera que si el panel está ubicado de manera fijo obtenemos una media de 4,10 voltios y si el panel está ubicado con el dispositivo de seguimiento obtenemos una media de 4,43 voltios.

$$\left(\frac{\mathbf{4,43}}{\mathbf{4,10}} - \mathbf{1}\right) * \mathbf{100} = \mathbf{7,91}\%$$

Por medio de este trabajo no se pretende comparar el rendimiento obtenido con los costes de realizar dicha mejora, pero dado que los materiales empleados para construir son bastante económicos, así para una pequeña instalación se podría aprovechar la máxima radiación posible.

Para que la generación de energía resultara beneficiosa por medio de esta mejora, se debería de minimizar el consumo de energía necesario para realizar todas las tareas de monitorización así de esta manera la energía empleada para ello no debe de suponer una pérdida de eficiencia porque de lo contrario llegaría a ser totalmente ineficiente este sistema.

6. Conclusiones

Con este trabajo se ha pretendido abordar la implementación de un prototipo con la finalidad de obtener una mejora en una instalación fotovoltaica gracias al uso de un soporte autoajustable el cual será construido con materiales “low cost”. De este modo es posible adaptar cualquier instalación particular existente siendo esta modificación mucho más económica que cualquier otro dispositivo ya comercializado en el mercado para este menester, debido a que los componentes requeridos para la modificación son fáciles de encontrar y son bastante económicos, necesitando únicamente de tiempo para su ensamblaje.

Por otra parte también se ha implementado un software encargado de monitorizar los dispositivos hardware, por el cual por medio de este software seremos capaces de llegar a obtener un histórico sobre la eficiencia conseguida al añadir esta mejora a los paneles solares respecto a una instalación fija. Este software desarrollado no solo se puede limitar a este único escenario planteado, sino que sería posible llevarlo a cualquier otro tipo de escenario realizando pequeñas modificaciones en el hardware empleado para monitorizar el panel solar. Este software sería posible emplearlo por ejemplo para monitorizar una instalación eólica, monitorizar una instalación eléctrica de una empresa, una vivienda, etc. De este modo por medio de este software sería posible tener un registro de la energía producida o consumida permitiendo hacer estudios sobre el histórico de datos obtenidos, o también se podrían llevar a cabo, previsiones en consumos o energía producida.

Gracias a la información obtenida y procesada sería posible hacer más eficiente una instalación eléctrica reduciendo los costes que esta pueda tener para por ejemplo una empresa consiguiendo que esta pueda llegar a ser más competitiva y siendo más respetuosa con el medio ambiente, o también beneficiando a los productores de energía consiguiendo optimizar su producción convirtiéndose en un beneficio económico.

Como conclusión final debemos incidir que el camino de las energías renovables está en marcha y es algo aceptado por la conciencia de todo el mundo, el cambio climático viene a consecuencia de la emisión de los gases combustibles, en donde la gran parte de ellos provenientes de la generación de energía, por esto mismo es hora de empezar a utilizar este tipo de energía como fuentes primarias, e ir delegando poco a poco a las energías contaminantes a un segundo puesto de consumo.

Es lógico suponer que el sector de las energías renovables experimentará un gran impulso en los próximos años, cuando se quiten ciertas barreras por parte de los gobiernos, con el consiguiente beneficio ecológico y al mismo tiempo abrirá un gran abanico de posibilidades a la industria de las energías renovables y a los inversores públicos o privados que apuesten por estos tipos de tecnología.

7. Trabajos Futuros

Durante el periodo de desarrollo de este trabajo han ido apareciendo observaciones, ideas y pruebas que han supuesto que se puedan plantear algunas mejoras al trabajo de manera que se podrían llegar a implementar, mejorando u optimizando de esta forma algunas facetas de este, convirtiéndolo en una versión más completa y estable. A continuación se van a mencionar algunas de las mejoras que se pueden añadir al trabajo:

7.1 Mejorar la conectividad de los paneles solares a la API por medio de una conexión inalámbrica.

Actualmente existe una limitación con el planteamiento propuesto en el trabajo, esta consiste en que la comunicación se realiza por medio de cable ya sea USB o RED. Esto puede suponer un hándicap a la hora de realizar ciertas instalaciones, debido a que el cableado puede incrementar el coste de la instalación además haciendo que la instalación pueda ser más compleja.

Por estos motivos sería conveniente dotar al dispositivo hardware de un módulo el cual le permita sustituta esa comunicación física, por una comunicación inalámbrica de manera que nos ahorraríamos la complejidad de tener un sistema de cableado.

7.2 Mejorar el portal web para visualizar más datos y añadir más funcionalidades

La implementación del portal web ha sido muy limitada debido a la falta de tiempo, pero para añadirle mucha más funcionalidad pueda ser un producto comercial, se debería de añadir alguna funcionalidad más como pueden ser: monitorización en tiempo real, sistema de alertas, control manual del panel, datos predictivos sobre consumos o producción, etc.

7.3 Mejorar el dispositivo hardware permitiendo crear alertas cuando se cumpla cierta condición programada.

Sería interesante añadir al dispositivo hardware de un sistema de alertas programables por las cuales se pueda obtener avisos de cuando alguno de ellos alcance el valor establecido. Un ejemplo claro seria controlar la temperatura del panel y cuando esta alcanzara cierto umbral, enviaría un mensaje avisando de este suceso, pudiéndose tomar medidas al respecto.

7.4 Mejorar el sistema de mensajes JSON utilizados para la comunicación en serial y red.

En la implementación de la comunicación por red se ha establecido de manera que envía un mensaje por cada valor obtenido de los sensores. Este método puede resultar ineficiente cuando se añaden varios dispositivos hardware a la red porque generarían demasiado tráfico. Por estos motivos sería más eficiente agrupar todos los mensajes obtenidos de los sensores en uno solo, de manera que solo se enviaría un único mensaje de mayor tamaño. De esta forma no se crearía tanto tráfico mejorando la congestión de la red.

7.5 Añadir un Bróker de mensajería para almacenar los mensajes.

Durante las pruebas reales realizadas, el prototipo ha tenido un comportamiento inesperado a la hora de enviar los mensajes JSON entre la API y el servidor central. Este comportamiento se

produjo porque cuando se realizaron las pruebas, se utilizó un cable de red con una longitud de 10 metros en vez del cable de 3 metros con el que se hicieron las pruebas de desarrollo. Este cable de tan larga longitud provocó que la red se saturase, haciendo que no fuera posible recibir mensajes en el servidor central. Se solucionó cambiando el cable a uno de longitud más reducida, pero esto no es una solución factible. Por este motivo una de las mejoras a considerar consisten en añadir un bróker de mensajería el cual nos permita ir almacenando los mensajes de manera que no se pueda perder ninguno e ir enviándolos al servidor central para su persistencia en base de datos cuando sea el mejor momento.

7.6 Modificar el sistema de seguimiento del panel solar, añadiendo valores de tablas existentes.

El sistema de seguimiento del panel ha sido implementado de manera que este calcula los valores de luminosidad en tiempo real, posicionando el panel de manera que este se orientará hacia donde mayor luminosidad obtenga. Este método tiene un pequeño inconveniente, el cual consiste en que hace uso de una pequeña cantidad de energía para hacer todo este procesamiento y posicionamiento dado que tiene que estar cada cierto tiempo obteniendo valores y calculando.

Existen tablas predefinidas como se muestra en la figura inferior en las cuales se especifican para cada estación del año la posición en la que estará el sol y su ángulo, pudiéndose de esta manera orientar el panel ajustándonos a los valores de dichas tablas. Para evitar situaciones conflictivas como pueden ser la aparición de nubes, sería interesante de dotar al dispositivo hardware de una combinación de ambas soluciones.

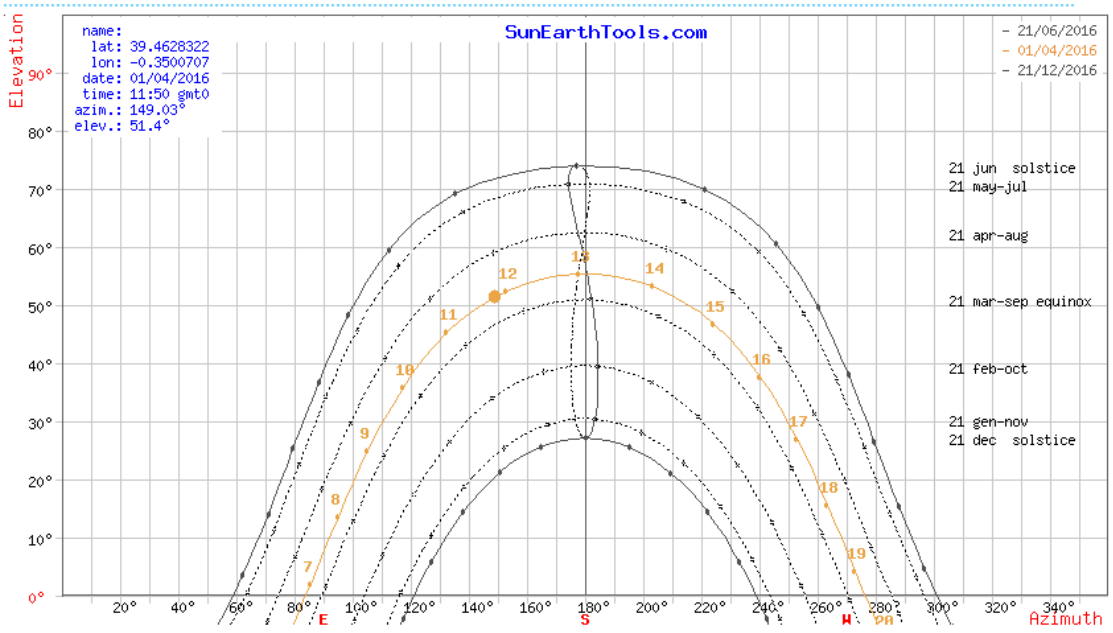


Fig. 6.1 Tabla de ángulos

En internet existen sitios web en donde es posible obtener las tablas mencionadas arriba, así como mapas con la representación visual de la posición del sol respecto a una posición establecida como se puede ver en la siguiente figura.

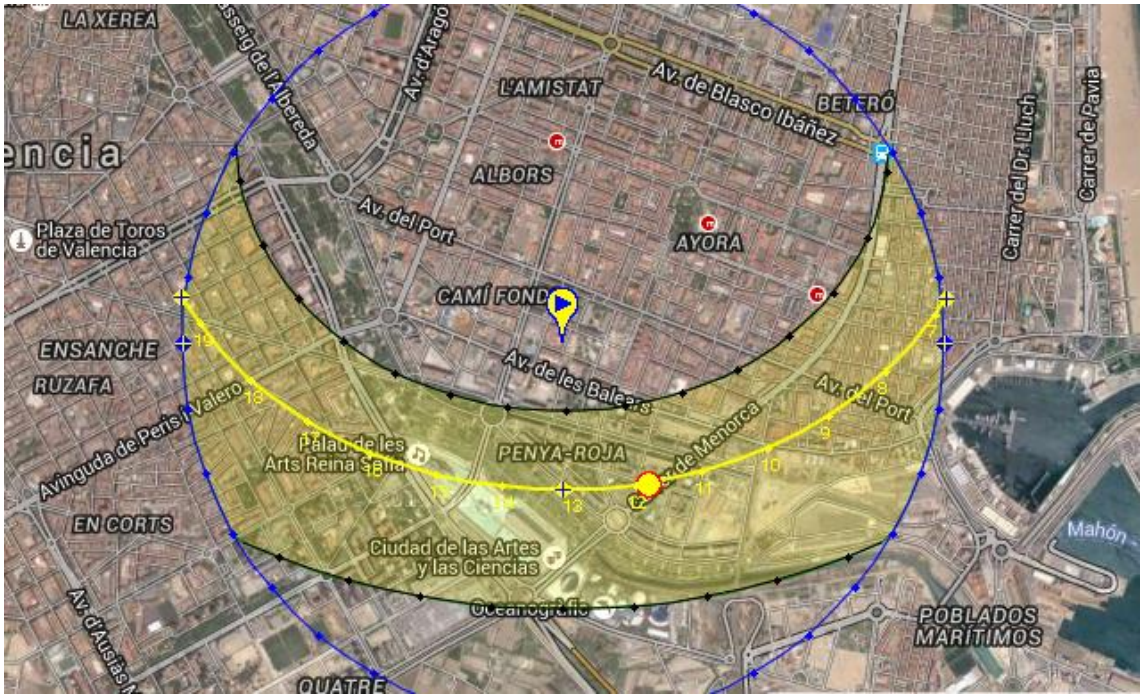


Fig. 6.2 Mapa con posición del sol respecto mi casa

7.7 Array de células fotovoltaicas

Una posible mejora muy eficiente sería la utilización de un array de células fotovoltaicas. Tendríamos nuestro dispositivo hardware adquiriendo datos de una sola placa solar del array, y a su vez transmitiría los datos para el correcto posicionamiento de la misma al resto. En donde el resto tendrían un dispositivo hardware más sencillo el cual únicamente permita el movimiento del panel.

Una vez adquiridos los datos de una de las placas, se podría extrapolar dicho resultado de control al resto de las placas, tan sólo teniendo en cuenta la diferente posición espacial que ocupan dichas placas con respecto a la que consideramos fuente de datos.

7.8 Autoaprendizaje del panel solar

Nuestro dispositivo hardware realiza una función de control en tiempo real, lo que conlleva que si un agente externo (como una nube) varía esos datos que adquiere (radiación solar) nuestro control sería desde este momento erróneo, siendo posible que llevara al artefacto a un estado de control irrecuperable. Sin embargo, una posible mejora podría ser un mecanismo de almacenamiento de históricos, de manera que, comparando datos de diferentes meses o estaciones del año, el propio dispositivo hardware pudiese llegar a extrapolar el movimiento del sol (velocidad, posición relativa).

Para ello se requerirían unas rutinas de cálculo bastantes más complejas que las utilizadas, además de un almacén de datos fácilmente sostenible y seguro.

8. Bibliografía

A continuación se van a detallar los recursos y las páginas web que han sido necesarios consultar durante el desarrollo del trabajo para llevarlo a cabo y solucionar dudas que han ido surgiendo durante su implementación.

[1.] Wikipedia (2016). Energía solar. *<https://es.wikipedia.org/wiki/Energ%C3%ADa_solar>*

[2.] Fondear, S.L. Todo sobre Controladores de Carga. *<http://www.fondear.org/infonautic/equipo_y_usos/Electricidad_Energia/ControladorCarga/ControladorCarga.htm>*

[3.] Sitio web oficial del framework AngularJs (2016). *<https://angularjs.org>*

[4.] Sitio web oficial de la plataforma libre Arduino (2016). *<https://forum.arduino.cc>*

[5.] Sitio web de programadores stackoverflow (2016). *<http://stackoverflow.com>*

[6.] Sitio web oficial WebService REST Jersey (2016). *https://jersey.java.net*

[7.] Sitio web oficial MyBatis (2016). *<http://www.mybatis.org/mybatis-3/es/>*

[8.] Sitio web oficial de Raspberry Pi Foundation (2016). *<https://www.raspberrypi.org>*

[9.] Sitio web oficial de bootstrap (2016). *<http://getbootstrap.com>*

[10.] Sitio web oficial Tomcat (2016). *<http://tomcat.apache.org>*

[11.] Sitio web oficial MySQL (2016). *<https://www.mysql.com>*

[12.] Sitio web oficial w3schools (2016). *<http://www.w3schools.com>*

[13.] Wikipedia (2016). JSON. *<https://es.wikipedia.org/wiki/JSON>*

[14.] Wikipedia (2016). Big Data. *<https://es.wikipedia.org/wiki/Big_data>*

[15.] Wikipedia (2016). Internet of Things. *<https://en.wikipedia.org/wiki/Internet_of_Things>*

[16.] Sitio web de SunEarthTools (2016). *<http://www.sunearthtools.com/dp/tools/pos_sun.php?lang=es>*

[17.] Alonso Javier Pérez Díaz. Tomcat, Apache Tomcat, Jakarta Tomcat. *<http://www.ajpdsoft.com/modules.php?name=Encyclopedia&op=content&tid=769>*

[18.] APS Valencia (2010). Eficiencia y rendimiento de un panel solar, 22 de febrero. *<http://apsvalencia.com/2010/02/22/eficiencia-y-rendimiento-de-un-panel-solar-certificaciones>*

[19.] Turrillas Solabre, Eduardo (2014). Estudio comparativo de la eficiencia energética en seguidores solares. Lugar de publicación: UPNA. *<http://academica-e.unavarra.es/xmlui/bitstream/handle/2454/11844/TFGTurrillasSalobreEduardo2014.pdf?sequence=1&isAllowed=y>*

[20.] Magaña Castañeda, Humberto (2010). DISEÑO DE UN SEGUIDOR SOLAR PARA EFICIENTAR LA CAPTACIÓN DE LA ENERGÍA SOLAR EN LOS PANELES FOTOVOLTAICOS.

Lugar de publicación: INSTITUTO TECNOLÓGICO SUPERIOR DE ARANDAS. *<http://www.tecarandas.edu.mx/descargas/seguidor_solar.pdf>*

[21.] Solano Martes, Loanny Francisca (2013). Análisis de los Sistemas de Gestión de Bases de Datos actuales como soporte para las tecnologías de Internet de las Cosas. Trabajo Final de Máster. Valencia: Universitat Politècnica de València. *<https://riunet.upv.es/bitstream/handle/10251/43325/Thesis%20Loanny%20Solano.pdf?sequence=1>*

[22.] Sánchez Chávez, Irma Yolanda (2014). Evaluación de Seguimiento Solar Acimutal para la Generación Fotovoltaica en el Centro de México. *<http://www.chi.itesm.mx/icm/wp-content/uploads/2014/12/1653548_Articulo_Seguidor_Eje_Acimutal.pdf>*