



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

Máster en Automática e Informática Industrial

Sistema de supervisión low-cost con algoritmo de diagnóstico predictivo de puntos calientes en paneles fotovoltaicos

Trabajo de final de máster

Autor: Mustapha Rouin Jarjour
Tutor : Emilio García Moreno

Curso: 2019-2020



Resumen

Uno de los principales y más frecuentes problemas de las instalaciones solares fotovoltaicas es la formación de puntos calientes, generalmente a causa de un sombreado parcial o suciedad, en los paneles solares.

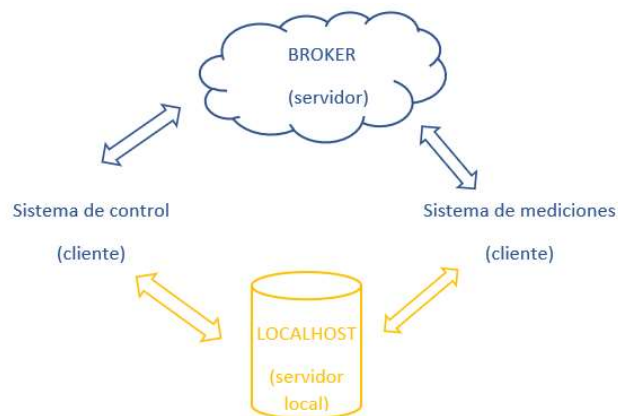
Este trabajo de final de máster pretende buscar una solución factible y óptima para este problema, comenzando por realizar simulaciones con la ayuda de la herramienta Simulink, para comprender mejor el comportamiento de los paneles solares y obtener modelos de funcionamiento de las celdas y paneles solares fotovoltaicos a partir de la irradiación solar y de la temperatura ambiente.

El siguiente paso es diseñar un sistema de lectura de mediciones de corriente y voltaje de los módulos solares individualmente, implementar un protocolo de comunicaciones entre el sistema de mediciones y el sistema de control y supervisión, escoger adecuadamente el sistema de control y supervisión y establecer correctamente la comunicación entre ambos sistemas.

Finalmente se desarrolla un algoritmo predictivo capaz de detectar una anomalía en las curvas de funcionamiento de los paneles solares, con el fin de frenar posibles apariciones de puntos calientes desde su fase inicial, tratando de evitar así que el fallo llegue a ser catastrófico.

Para implementar la idea descrita, se hace uso de un dispositivo cuyo software es gratuito, se trata de Arduino UNO, que se le agrega un módulo de expansión de entradas analógicas (29 entradas en total) para conectar los sensores de corriente y de voltaje de hasta 14 paneles fotovoltaicos por cada dispositivo Arduino, y una tarjeta de red para conectar el dispositivo a la red de Internet mediante un cable Ethernet.

La supervisión y monitorización se lleva a cabo mediante Codesys, un entorno gratuito de desarrollo para la programación de controladores conforme con el estándar industrial internacional IEC 61131-3. Codesys ofrece la opción de utilizar un PLC virtual que se instala en el PC. Para el caso de este trabajo, se hará uso de esta herramienta. Y por último, el protocolo de comunicaciones escogido es el actual MQTT enfocado a IoT.



Palabras clave

Energía solar, célula fotovoltaica, panel fotovoltaico, puntos calientes, Arduino, MQTT, broker, topic, Ethernet, Codesys, supervisión, monitorización, IoT, PLC, PC.

Resum

Un dels principals i més freqüents problemes de les instal·lacions solars fotovoltaïques és la formació de punts calents, generalment a causa d'un sombreatge parcial o brutícia, en els panells solars.

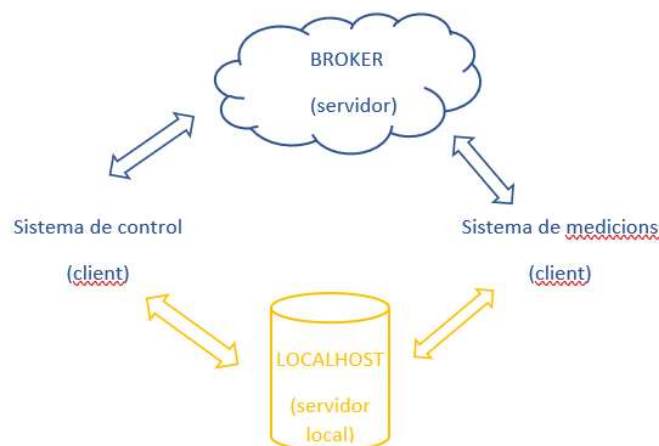
Aquest treball de fi de màster pretén buscar una solució factible i òptima per a aquest problema, començant per realitzar simulacions amb l'ajuda de l'eina Simulink, per comprendre millor el comportament dels panells solars i obtenir models de funcionament de les cel·les i panells solars fotovoltaïcs a partir de la irradiació solar i de la temperatura ambient.

El següent pas és dissenyar un sistema de lectura de mesuraments de corrent i voltatge dels mòduls solars individualment, implementar un protocol de comunicacions entre el sistema de mesuraments i el sistema de control i supervisió, escollir adequadament el sistema de control i supervisió i establir correctament la comunicació entre els dos sistemes.

Finalment es desenvolupa un algoritme predictiu capaç de detectar una anomalia en les corbes de funcionament dels panells solars, per tal de frenar possibles aparicions de punts calents des de la seva fase inicial, tractant d'evitar així que la decisió arribi a ser catastròfic.

Per implementar la idea descrita, es fa ús d'un dispositiu que el software de programació del mateix és gratuït, es tracta d'Arduino UNO, que se li afegeix un mòdul d'expansió d'entrades analògiques (29 entrades en total) per connectar els sensors de corrent i de voltatge de fins a 14 panells fotovoltaïcs per cada dispositiu Arduino, i una targeta de xarxa per connectar el dispositiu a la xarxa d'Internet mitjançant un cable Ethernet.

La supervisió i monitorització es porta a terme mitjançant CoDeSys, un entorn gratuït de desenvolupament per a la programació de controladors d'acord amb l'estàndard internacional IEC 61131-3. CoDeSys ofereix l'opció d'utilitzar un PLC virtual que s'instal·la al PC. Per al cas d'aquest treball, es farà ús d'aquesta eina. I finalment, el protocol de comunicacions escollit és l'actual MQTT enfocat a IoT.



Paraules clau

Energia solar, cèl·lula fotovoltaïca, panell fotovoltaïc, punts calents, Arduino, MQTT, broker, topic, Ethernet, CoDeSys, supervisió, monitorització, IOT, PLC, PC.

Abstract

One of the main and most frequent problems with solar photovoltaic installations is the formation of hot spots, usually due to partial shading or dirt, on the solar panels.

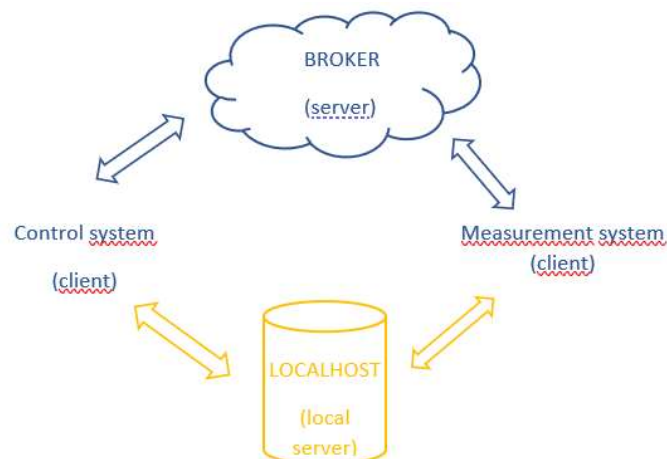
This final master's work aims to find a feasible and optimal solution to this problem, starting by performing simulations with the help of the Simulink tool, to better understand the behavior of solar panels and to obtain models of the functioning of photovoltaic solar cells and panels from solar irradiation and ambient temperature.

The next step is to design a system for reading current and voltage measurements of the individual solar modules, implement a communication protocol between the measurement system and the control and monitoring system, choose the control and monitoring system appropriately, and establish correct communication between the two systems.

Finally, a predictive algorithm capable of detecting an anomaly in the operating curves of the solar panels is developed, in order to stop possible hot spots from appearing from their initial phase, thus trying to prevent the failure from becoming catastrophic.

To implement the idea described, a device is used whose software is free, it is Arduino UNO, which adds an expansion module of analog inputs (29 inputs in total) to connect the current and voltage sensors of up to 14 photovoltaic panels for each Arduino device, and a network card to connect the device to the Internet through an Ethernet cable.

The supervision and monitoring are carried out by Codesys, a free development environment for the programming of controllers according to the international industrial standard IEC 61131-3. Codesys offers the option of using a virtual PLC that is installed in the PC. For the case of this work, this tool will be used. And finally, the communication protocol chosen is the current MQTT focused on IoT.



Keywords

Solar energy, photovoltaic cell, photovoltaic panel, hot spots, Arduino, MQTT, broker, topic, Ethernet, Codesys, supervision, monitoring, IoT, PLC, PC



Índice

Resumen.....	3
Palabras clave.....	3
Resum.....	4
Paraules clau	4
Abstract	5
Keywords.....	5
Índice.....	7
Índice de figuras	9
1. Introducción	11
2. Motivación	12
3. Objetivos	12
4. Marco teórico.....	13
4.1. Energía solar fotovoltaica.....	13
4.2. Célula fotovoltaica.....	14
4.3. Fenómeno fotovoltaico	14
4.4. Modelo eléctrico y parámetros que definen la célula fotovoltaica	15
4.5. Modelo de la célula fotovoltaica en Simulink	17
4.6. Efecto de la irradiancia en la célula fotovoltaica (a 25°C)	18
4.7. Efecto de la temperatura en la célula fotovoltaica (a 1000 W/m ²)	19
4.8. Panel fotovoltaico	21
4.9. Fenómeno <i>hot spot</i>	22
5. Impacto Esperado	23
6. Estado del arte	24
7. Identificación y análisis de soluciones posibles	25
8. Solución propuesta.....	26
8.1. Hardware y software utilizados.....	27
8.2. Diseño de la solución propuesta	33
8.3. Desarrollo de la solución propuesta.....	35
8.4. Implantación.....	38
9. Presupuesto	46
10. Conclusiones.....	47
11. Trabajos futuros	48
12. Referencias.....	49



Anexo I: Códigos y programas.....	51
Código programa Arduino.....	51
Programa Codesys.....	56
Anexo II: Información técnica del hardware utilizado	60
Arduino UNO	60
Ethernet Shield W5100	60
Shield expansión entradas analógicas Valleman	61
Transductor de voltaje CELSA DMU	62
Transductor de corriente CELSA DMI.....	63

Índice de figuras

Figura 1: Potencia instalada FV España (Fuente datos: Red Eléctrica de España)	11
Figura 2: Célula fotovoltaica silicio monocristalino.....	14
Figura 3: Unión PN	15
Figura 4: Circuitos equivalentes de una célula fotovoltaica	15
Figura 5: Diagrama de Simulink para la célula fotovoltaica	17
Figura 6: Curva Voltaje - Corriente a 1000 W/m ² y 25 °C	17
Figura 7: Curva Voltaje – Potencia a 1000 W/m ² y 25 °C.....	18
Figura 8: Efecto de la irradiancia en la curva Voltaje - Corriente (a 25°C).....	18
Figura 9: Efecto de la irradiancia en la curva Voltaje – Potencia (a 25°C)	19
Figura 10: Efecto de la temperatura en la curva Voltaje - Corriente (a 1000 W/m ²)	20
Figura 11: Efecto de la temperatura en la curva Voltaje – Potencia (a 1000 W/m ²).....	20
Figura 12: Efecto de la irradiancia en la curva V – I panel FV (a 25°C)	21
Figura 13: Efecto de la irradiancia en la curva V – P panel FV (a 25°C)	21
Figura 14: Efecto de la temperatura en la curva V – I panel FV (a 1000 W/m ²)	22
Figura 15: Efecto de la temperatura en la curva V – P panel FV (a 1000 W/m ²)	22
Figura 16: Daños puntos calientes	23
Figura 17: Esquema general comunicaciones MQTT	26
Figura 18: Placa Arduino UNO.....	27
Figura 19: Expansión entradas analógicas Velleman_KA12.....	27
Figura 20: Tarjeta de red Ethernet W5100	28
Figura 21: Transductor de voltaje	28
Figura 22: Transductor de corriente	29
Figura 23: Arduino IDE (entorno de desarrollo integrado)	29
Figura 24: Entorno de desarrollo Codesys	30
Figura 25: Software CODESYS Control Win SL.....	31
Figura 26: Software CODESYS SoftMotion	31
Figura 27: CODESYS Gateway.....	32
Figura 28: Broker shiftr.io en la nube.....	32
Figura 29: Broker shiftr.io en el PC (aplicación de escritorio).....	33
Figura 30: Sistema de mediciones.....	33
Figura 31: Transductores de corriente y de voltaje	34
Figura 32: Ventana general de monitorización.....	34
Figura 33: Esquema general sistemas de control y de mediciones	35
Figura 34: Lectura de voltaje y de corriente, cálculo de la potencia y publicación de ésta en el broker.....	36
Figura 35: Sistema de mediciones publicando las potencias de los paneles en el broker shiftr.io	37
Figura 36: Sistema de control suscrito a los topics donde publica el sistema de mediciones y recibiendo los mensajes de dichos topics.....	37
Figura 37: Tabla pines de conexión de los sensores	38
Figura 38: Código Arduino lectura sensores, conversión valores, cálculo potencia y publicación en el broker	39
Figura 39: Librerías necesarias Arduino	39
Figura 40: Configuración broker en Arduino.....	39
Figura 41: Código Arduino para declaración topics	40



Figura 42: Variables a definir por cada panel en Codesys.....	40
Figura 43: Bloque Cliente MQTT Codesys	41
Figura 44: Declaración variables de entrada.....	41
Figura 45: Ventana general de monitorización en estado reposo	42
Figura 46: Ventana general de monitorización en estado activo	42
Figura 47: Ventana general de monitorización zona inferior	43
Figura 48: Gestor de alarmas: Alarmas activas	43
Figura 49: Gestor de alarmas: Historial alarmas	44
Figura 50: Ventana individual de monitorización	45
Figura 51: Ventana individual de monitorización zona inferior.....	45
Figura 52: Desglose presupuesto	46
Figura 53: Bloques MQTT_Client Codesys.....	58
Figura 54: Convertir mensaje de texto leído (REAL) a dato numérico (REAL)	59
Figura 55: Configurar gráficas	59
Figura 56: Descripción tarjeta Arduino UNO.....	60
Figura 57: Tarjeta de red Ethernet W5100	61
Figura 58: Descripción módulo expansión entradas analógicas	62
Figura 59: Especificaciones transductor de voltaje.....	62
Figura 60: Especificaciones transductor de corriente.....	63

1. Introducció

Entre 2006 y 2009 desde el gobierno se desarrolló una campaña que animaba a participar en proyectos fotovoltaicos, un atractivo sistema de primas para las renovables vía Decreto. Gracias a esta campaña gubernamental, España pasó de tener 0 potencia fotovoltaica a más de 3.300 megavatios en 2008.

Este auge por la energía fotovoltaica tan solo ha durado hasta 2010, cuando empezaron los recortes a las primas prometidas a causa de la inmersión del país en la crisis económica. Estos recortes no solamente se afianzaron durante los siguientes años, sino que, además, el gobierno de turno impulsó el controvertido 'Impuesto al Sol'. La fotovoltaica había quedado paralizada.

No fue hasta el año 2017 que la energía solar empezó a resucitar en España, gracias a una nueva regulación que levantó el llamado 'impuesto al sol', y gracias también a la enorme reducción del precio de los paneles fotovoltaicos. El precio se ha reducido en un 95% en los últimos 15 años.

Solo en apenas 10 meses (enero a octubre de 2019) se instalaron en España más nuevos megavatios de potencia fotovoltaica que en los últimos diez años. Según datos de la patronal del sector en Europa, España plantó el 25% de todos los MWs de fotovoltaica del viejo continente en 2019.

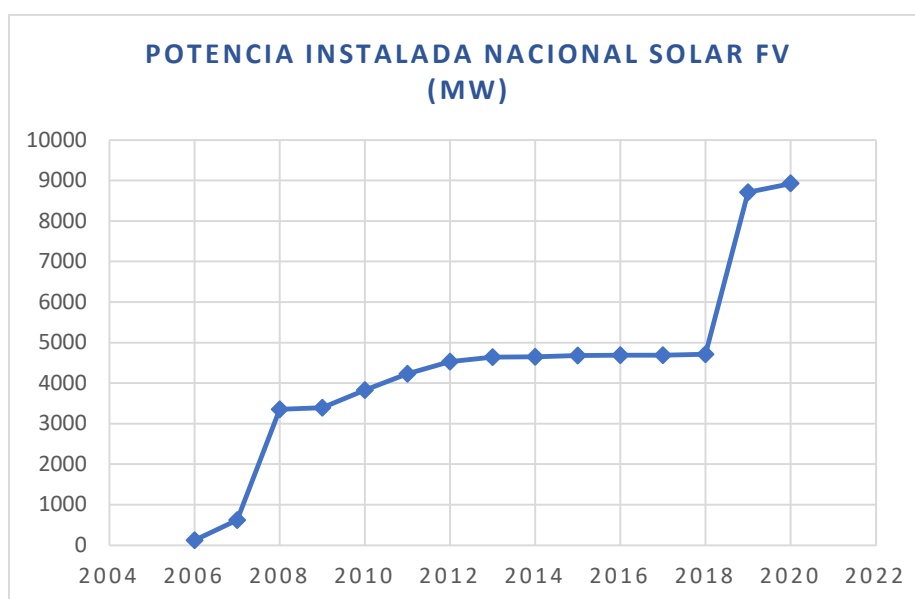


Figura 1: Potencia instalada FV España (Fuente datos: Red Eléctrica de España)

La energía solar fotovoltaica procede de una fuente limpia e inagotable, el sol, y tiene muchas ventajas con respecto a las demás fuentes de energía, sobre todo, las convencionales o no renovables. Sin embargo, también tiene desventajas frente a las fuentes convencionales, como por ejemplo la eficiencia de los módulos fotovoltaicos que no supera el 21% en el mejor caso para los módulos convencionales de silicio monocristalino, o la posible aparición de puntos calientes en determinadas celdas de los paneles solares, cuyo efecto puede variar desde la

disminución en la cantidad de potencia que puede aportar el módulo fotovoltaico hasta el deterioro total de los módulos afectados.

Este proyecto pretende analizar la problemática de la aparición de puntos calientes en los módulos fotovoltaicos, para posteriormente, diseñar e implementar un algoritmo predictivo que permita detectar la aparición de puntos calientes en su fase inicial, para tratar de evitar así que la anomalía llegue a alcanzar el estado catastrófico.

2. Motivación

La histórica preocupación por el medio ambiente, junto con las medidas tomadas por el gobierno para fomentar la producción de energía solar, y el desplome de los precios de los paneles fotovoltaicos, permitieron un auge en el sector de la energía fotovoltaica en los últimos dos años. En la actualidad, la potencia instalada de energías renovables representa más del 45% del total nacional, la fotovoltaica, más del 8% (datos del año 2019).

Uno de los problemas más importantes a resolver en los sistemas de producción de energía fotovoltaica son los puntos calientes que pueden aparecer en determinadas zonas de los módulos fotovoltaicos a causa de sombreado parcial o suciedad entre otros factores, y conlleva a la disminución de la energía producida por el panel o paneles afectados y a la degradación del propio módulo fotovoltaico.

Trabajar en solventar esta problemática podría aumentar la generación y alargar la vida útil de los módulos fotovoltaicos. Esto consolidaría la idea de impulsar la energía solar.

3. Objetivos

El objetivo general del proyecto es el diseño y desarrollo de un sistema para la supervisión de paneles solares fotovoltaicos para detectar anomalías en las curvas de generación fotovoltaica, con el fin de frenar posibles apariciones de puntos calientes desde su fase inicial, tratando de evitar así que el fallo llegue a ser catastrófico. Para ello se definen los siguientes objetivos específicos:

- Estudiar el efecto fotovoltaico.
- Simular el funcionamiento de una célula fotovoltaica.
- Obtener un modelo de las curvas de funcionamiento ideales.
- Analizar el efecto 'hot spot' (punto caliente).
- Simular el efecto de los puntos calientes y comprender el comportamiento de este fenómeno.
- Diseñar un sistema para la medición en tiempo real de la corriente y la tensión de cada panel.
- Seleccionar un medio de comunicación para enlazar el sistema de medición con el sistema de control y monitorización.
- Escoger un protocolo de comunicación adecuado para la transmisión de los datos.
- Desarrollar un algoritmo predictivo para la detección de puntos calientes a partir de las mediciones de corriente y tensión de los paneles.

- Diseñar un sistema de Supervisión, Control y Adquisición de Datos (SCADA) para la monitorización del sistema en tiempo real.
- Programar alarmas para notificar posibles situaciones de posible aparición de puntos calientes.
- Ofrecer la posibilidad de generar archivos con historiales de la generación fotovoltaica de los distintos paneles.
- Ofrecer la posibilidad de generar archivos con historiales de alarmas.
- Ofrecer la posibilidad de actuar sobre el sistema de mediciones desde el sistema de control y monitorización.

4. Marco teórico

4.1. Energía solar fotovoltaica

La energía solar fotovoltaica permite producir electricidad de origen renovable transformando de manera directa y limpia la luz solar en energía eléctrica, basándose en la teoría del efecto fotovoltaico. Al incidir la radiación del sol sobre una de las caras de una célula fotoeléctrica, se produce una diferencia de potencial entre ambas caras que hace que los electrones salten de un lugar a otro, generando así corriente eléctrica.

A diferencia de las fuentes de energía convencionales, la energía solar fotovoltaica no agota su fuente primaria (el sol) tras transformar su energía en energía eléctrica. Además, esta fuente primaria no requiere de ningún tratamiento previo por el hombre, ni deja residuos contaminantes.

Las instalaciones solares fotovoltaicas pueden ser básicamente de dos tipos: instalaciones aisladas, orientadas fundamentalmente a instalaciones de bombeo, señalización, comunicaciones y electrificación rural, e instalaciones conectadas a red, orientadas a la venta de energía eléctrica y autoconsumo.

Las posibilidades de aplicación de la energía solar fotovoltaica son inmensas y abarcan desde las más simples como calculadoras y relojes solares, a las más complejas como grandes plantas de generación eléctrica o sistemas de alimentación para satélites artificiales.

Algunas de las ventajas de utilizar la energía solar fotovoltaica y el resto de las energías renovables en general son:

- Evitar las grandes emisiones de CO₂.
- Contribuir a la mejora del medio ambiente.
- Producción de energía de forma descentralizada.
- Aumenta la independencia energética.
- Mejorar la calidad del servicio eléctrico.
- Reducir el consumo de petróleo y otros tipos de combustibles contaminantes y que no abundan en el territorio nacional.
- Permite generar energía eléctrica en lugares de difícil electrificación.

4.2. Célula fotovoltaica

La célula o celda fotovoltaica, es el elemento encargado de transformar la energía luminosa en energía eléctrica mediante el efecto fotovoltaico. Las células fotovoltaicas están compuestas de un material semiconductor, generalmente silicio puro con adición de impurezas de ciertos elementos químicos (boro y fósforo habitualmente).

La eficiencia de conversión media obtenida por las células disponibles comercialmente está alrededor del 11-12%, que varía según la tecnología utilizada, desde el 6% para las células de silicio amorfo hasta el 14-21% para las células de silicio monocristalino.



Figura 2: Célula fotovoltaica silicio monocristalino

Dadas las características eléctricas que podemos obtener de las células fotovoltaicas, que tenemos bajas tensiones (en torno a 0.6 voltios) y corrientes que pueden alcanzar hasta los 10 amperios, vamos a necesitar combinar muchas células, conectándolas en serie, formando así el panel fotovoltaico, para aumentar el voltaje del sistema, manteniendo la corriente.

4.3. Fenómeno fotovoltaico

El fenómeno fotovoltaico fue descubierto en 1839 por el científico francés, Henri Becquerel. Las primeras celdas solares de selenio fueron desarrolladas en 1880, sin embargo, no fue sino hasta 1950 que se desarrollaron las celdas de silicio monocristalino que actualmente dominan la industria fotovoltaica.

En los semiconductores de las células solares (generalmente silicio monocristalino), para crear la existencia de movimiento de electrones en una única dirección, se crea un campo eléctrico permanente entre ambas caras de la célula, a través de uniones PN.

La región N se dopa con fósforo. El fósforo tiene cinco electrones de valencia, uno más que el silicio, de modo que esta región muestra una afinidad por los electrones menor que el silicio. La otra región, denominada P, se dopa con boro. El boro tiene sólo tres electrones de valencia, por lo que su afinidad para captar electrones es mayor que la del silicio puro. Al recibir irradiación solar, los electrones del silicio se excitan y se propagan entre ellos, además, cuando nos

encontramos en el caso de dopaje PN, los electrones fluyen desde la capa con mayor número de electrones hacia la capa con menor número de éstos.

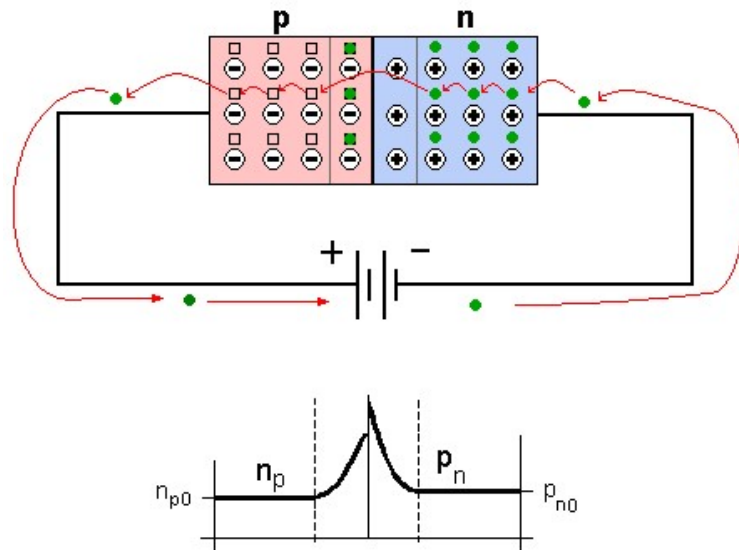


Figura 3: Unión PN

Sin embargo, la generación eléctrica de una celda es muy baja, inservible para alimentar una carga genérica, esto conlleva a la aparición del panel fotovoltaico. Los paneles o módulos fotovoltaicos están formados por un conjunto de este tipo de células interconectadas entre ellas, permitiendo así producir mayor cantidad de energía eléctrica a partir de la luz que incide sobre ellos mediante el efecto fotoeléctrico.

4.4. Modelo eléctrico y parámetros que definen la célula fotovoltaica

La celda fotovoltaica es el elemento más importante en la generación de energía fotovoltaica, cuyo modelo eléctrico se corresponde con la ilustración de la figura 4. Generalmente se modela como una fuente de corriente ideal, en paralelo con un diodo que representa el comportamiento físico de la unión PN, y para reflejar el efecto de un panel real (que conlleva la aparición de unas corrientes parásitas), se introducen las resistencias R_p y R_s .

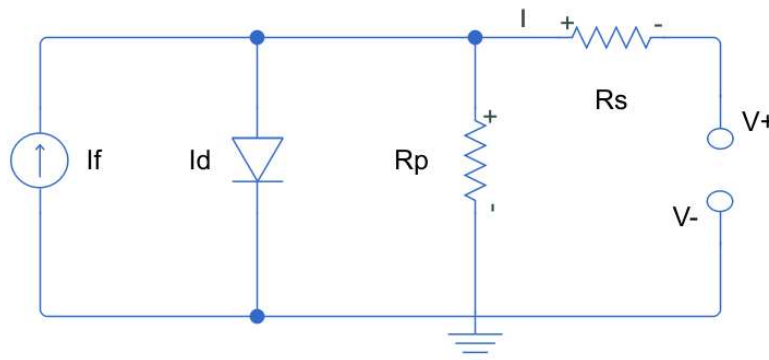


Figura 4: Circuitos equivalentes de una célula fotovoltaica

La resistencia serie R_s representa el contacto entre la estructura metálica y la capa del semiconductor tipo N. Esta resistencia provoca una disminución en el voltaje de salida, que a su vez tiene un impacto directo en la potencia.

La resistencia en paralelo R_p representa la corriente de fuga que se produce en la unión PN debido a las impurezas. Esta resistencia afecta en menor proporción a la pérdida de potencia en la célula fotovoltaica.

$$I = I_f(G, T) - N_p * I_d * \left[e^{\left(\frac{V+I*R_s}{V_t} - 1\right)} \right] - \frac{V+I*R_s}{R_p} \quad (1)$$

$$V_t = m * N_s * k * \frac{(T+273)}{q e} \quad (2)$$

$$I_d = \frac{I_{sc} - \frac{V_{oc}}{R_p}}{e^{\left(\frac{V_{oc}}{V_t} - 1\right)}} \quad (3)$$

Dónde:

- **N_s** es el número de celdas en serie del panel solar.
- **N_p** es el número de celdas en paralelo.
- **k** es la constante de Boltzman.
- **q_e** es la carga del electrón.
- **m** es el factor de idealidad del diodo ($1 < m < 2$).
- **T** es la temperatura de trabajo del panel solar en °C.
- **G** es la irradiancia en W/m²
- **R_s** es la resistencia serie.
- **R_p** es la resistencia en paralelo.
- **I_f** es la corriente fotogenerada (depende directamente de la irradiancia y de la temperatura).
- **I_{sc}** es la corriente de cortocircuito (aumenta con la radiación incidente).
- **I_d** es la corriente inversa de saturación del diodo.
- **V_{oc}** es la tensión de circuito abierto.

Para utilizar la ecuación (1), es necesario conocer el valor de las resistencias R_s y R_p , parámetros característicos de cada panel y el factor de idealidad del diodo (m). Estos parámetros están relacionados con el material empleado en su fabricación y en general se determinan de forma experimental ya que el fabricante no provee esta información. Una vez calculados se consideran constantes en todo el rango de operación y para cualquier valor de irradiancia y temperatura.

Para aumentar la precisión en los cálculos, en la ecuación (1) se introduce la dependencia de la corriente de cortocircuito y de la tensión de circuito abierto, del valor de la irradiancia solar y de la temperatura de operación del panel con relación a las condiciones estándar dadas por el fabricante ($T_{std} = 25$ °C, $G_{std} = 1000$ W/m²).

$$I_{sc}(G, T) = N_p * I_{sc} * \frac{G}{G_{std}} + \alpha * I_{sc}(T - T_{std}) \quad (4)$$

$$V_{oc}(G, T) = V_{oc} + V_t * \ln \left[\frac{G}{G_{std}} \right] + \beta * (T - T_{std}) \quad (5)$$

4.5. Modelo de la célula fotovoltaica en Simulink

Para observar el comportamiento general de la celda fotovoltaica se ha implementado un modelo con la ayuda de las herramientas Matlab y Simulink. La siguiente ilustración muestra el diagrama en Simulink:

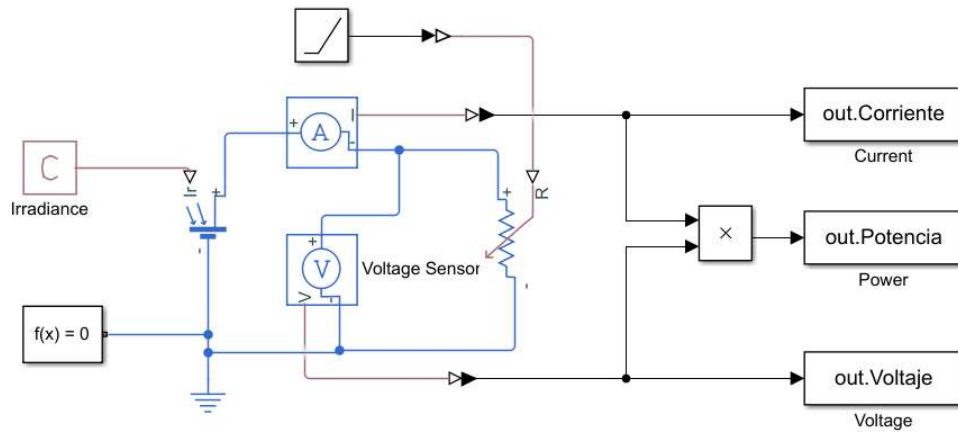


Figura 5: Diagrama de Simulink para la célula fotovoltaica

La manera más habitual para definir el comportamiento de las celdas y paneles fotovoltaicos es mediante las gráficas Voltaje – Corriente y Voltaje – Potencia, también llamadas curvas características. Se ha desarrollado un script de Matlab para iniciar las simulaciones, recoger los datos, tratarlos y graficar los resultados. La siguiente ilustración muestra el comportamiento de la célula fotovoltaica en condiciones estándar ($T_{std}= 25\text{ }^{\circ}\text{C}$, $G_{std}=1000\text{ W/m}^2$):

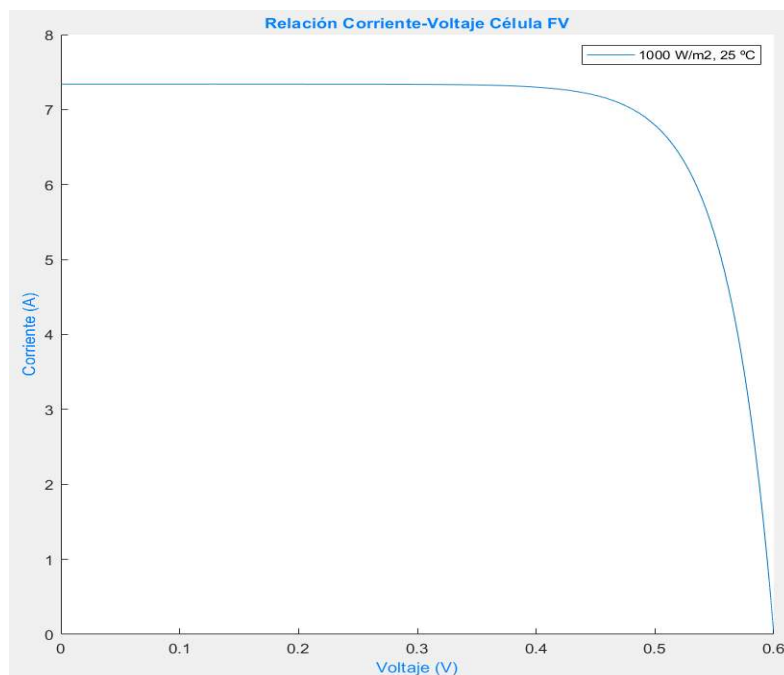


Figura 6: Curva Voltaje - Corriente a 1000 W/m^2 y $25\text{ }^{\circ}\text{C}$

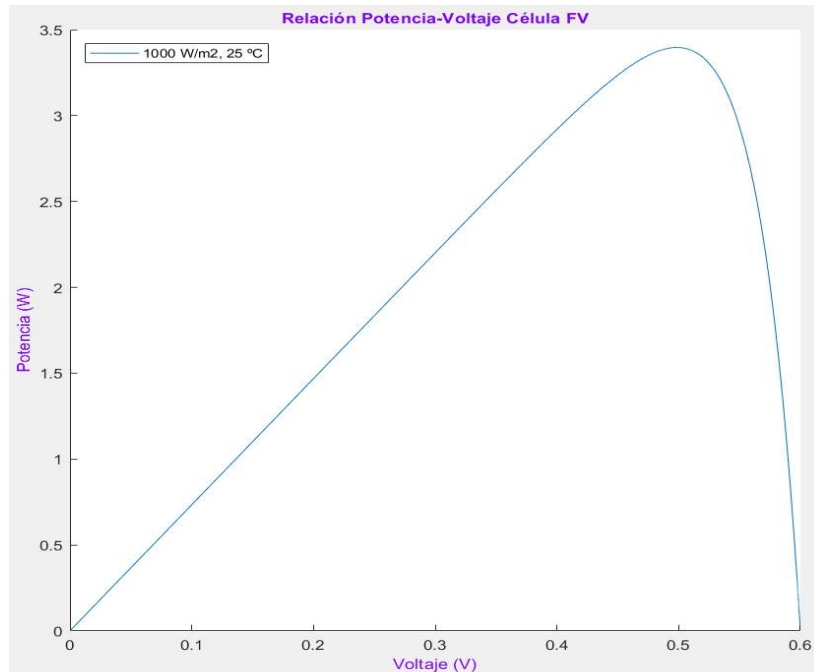


Figura 7: Curva Voltaje – Potencia a 1000 W/m² y 25 °C

4.6. Efecto de la irradiancia en la célula fotovoltaica (a 25°C)

La corriente de salida de la celda fotovoltaica es directamente proporcional a la irradiancia que recibe, de forma que, si disminuye la irradiancia solar, disminuye también la corriente en la misma proporción. Por otro lado, la tensión también varía ante variaciones en la irradiancia, pero en mucho menor medida:

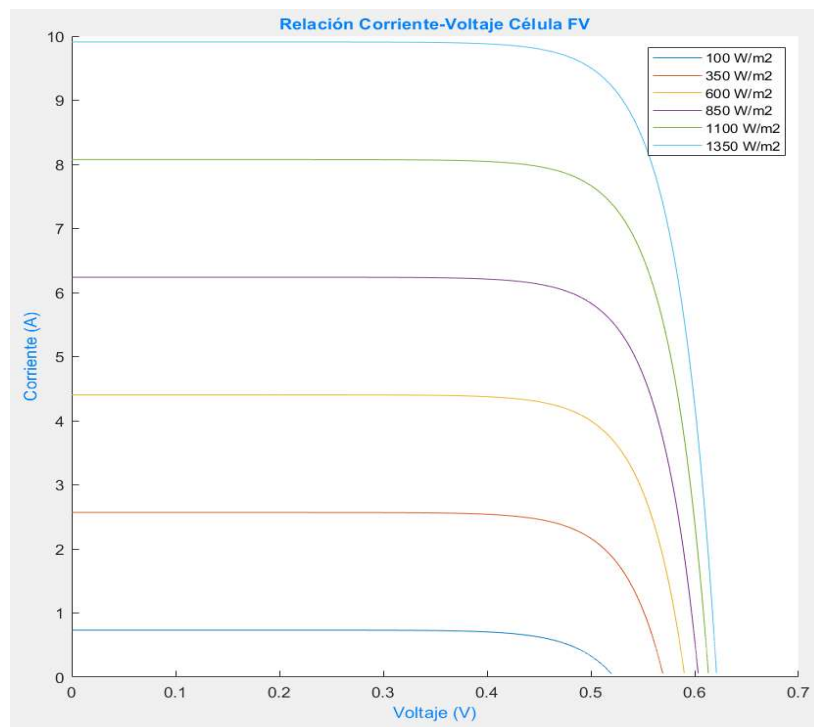


Figura 8: Efecto de la irradiancia en la curva Voltaje - Corriente (a 25°C)

Como consecuencia de estas variaciones, especialmente por la variación de la corriente, también varía la potencia entregada por la célula fotovoltaica:

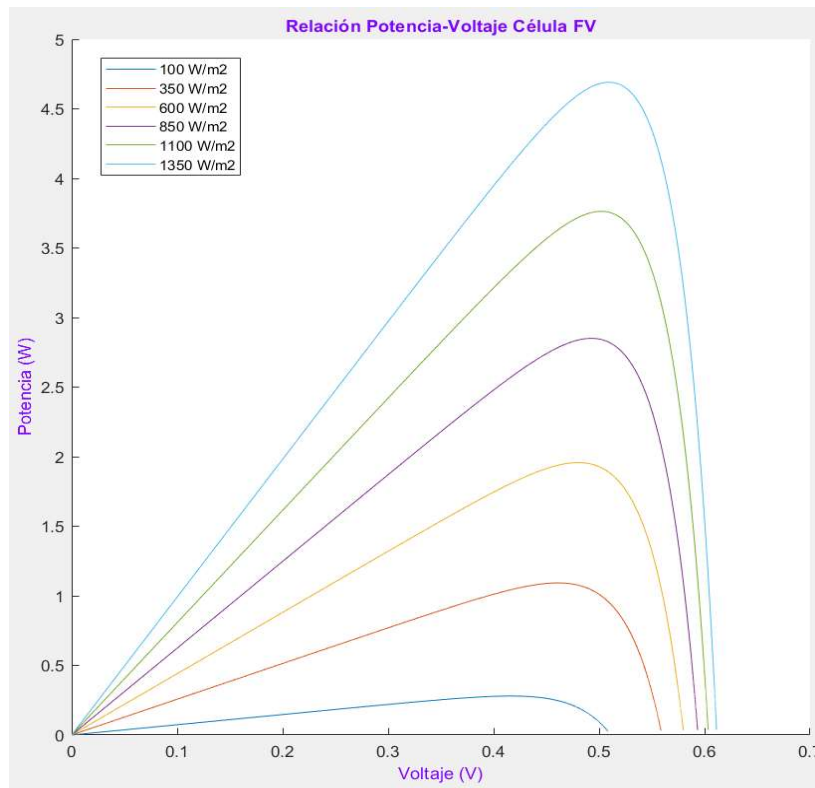


Figura 9: Efecto de la irradiancia en la curva Voltaje – Potencia (a 25°C)

4.7. Efecto de la temperatura en la célula fotovoltaica (a 1000 W/m²)

La temperatura de la célula afecta al valor de la tensión en circuito abierto, de manera que, si aumenta la temperatura, la tensión de circuito abierto disminuye del orden de unos pocos milivoltios por cada grado centígrado que aumenta la temperatura (2,3 mV/°C para el silicio):

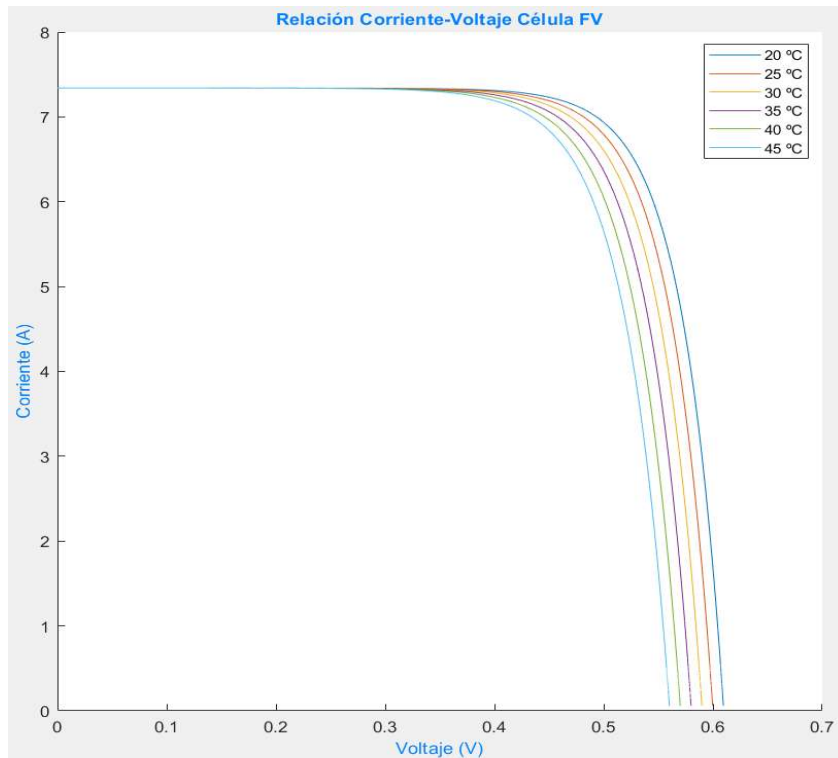


Figura 10: Efecto de la temperatura en la curva Voltaje - Corriente (a 1000 W/m²)

Como consecuencia de la disminución del voltaje a raíz del aumento de la temperatura, también varía la potencia entregada por la célula fotovoltaica, aunque en mucho menor proporción que en el caso de la disminución de la irradiancia solar:

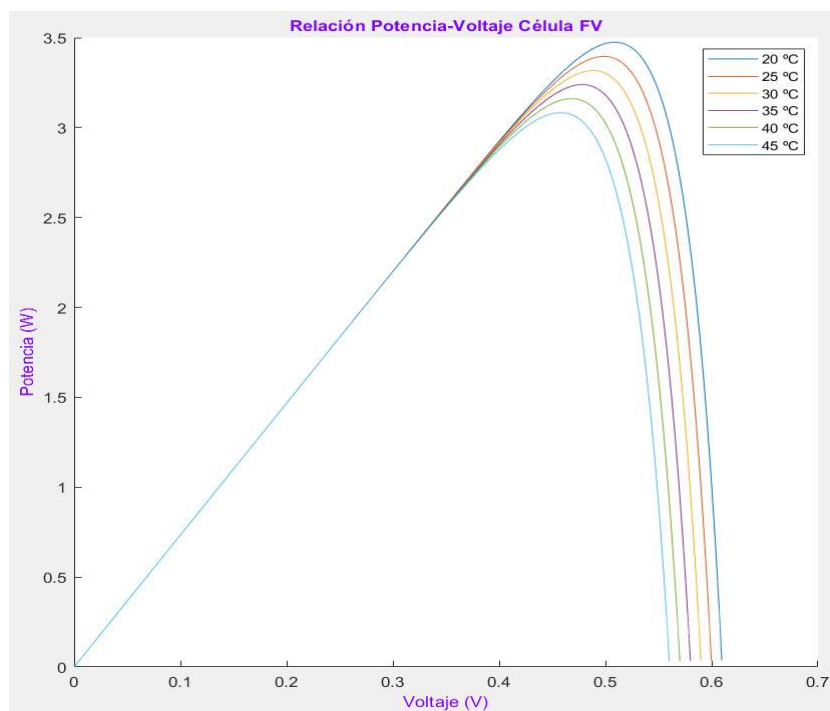


Figura 11: Efecto de la temperatura en la curva Voltaje – Potencia (a 1000 W/m²)

4.8. Panel fotovoltaico

El módulo o panel fotovoltaico es el resultado de la interconexión de muchas células solares fotovoltaicas en serie y en paralelo para obtener unos valores determinados de tensión y de corriente.

El conjunto de células están interconectadas eléctricamente, encapsuladas, y montadas en una estructura de soporte o marco. Esta unión de células permite obtener potencias considerables y voltajes relativamente altos.

A continuación, se ilustran las gráficas de las curvas características de un panel genérico de silicio monocristalino de 225 W y sus dependencias con la irradiancia solar y la temperatura respectivamente:

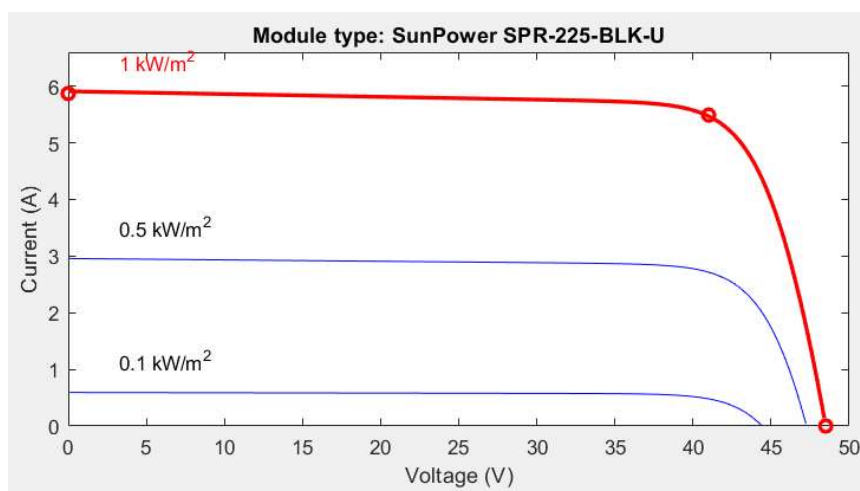


Figura 12: Efecto de la irradiancia en la curva $V-I$ panel FV (a 25°C)

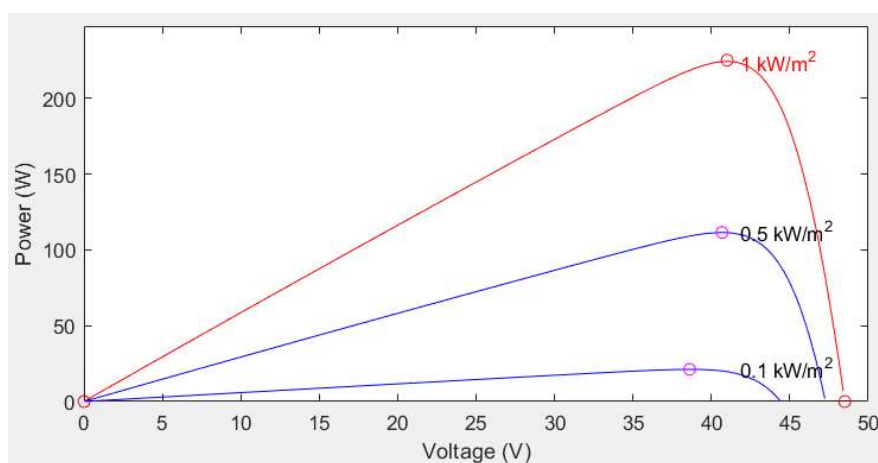


Figura 13: Efecto de la irradiancia en la curva $V-P$ panel FV (a 25°C)

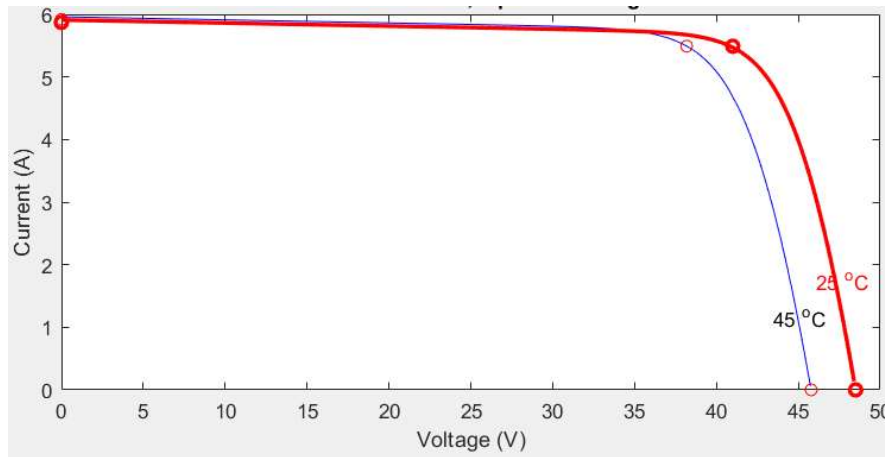


Figura 14: Efecto de la temperatura en la curva $V - I$ panel FV (a 1000 W/m^2)

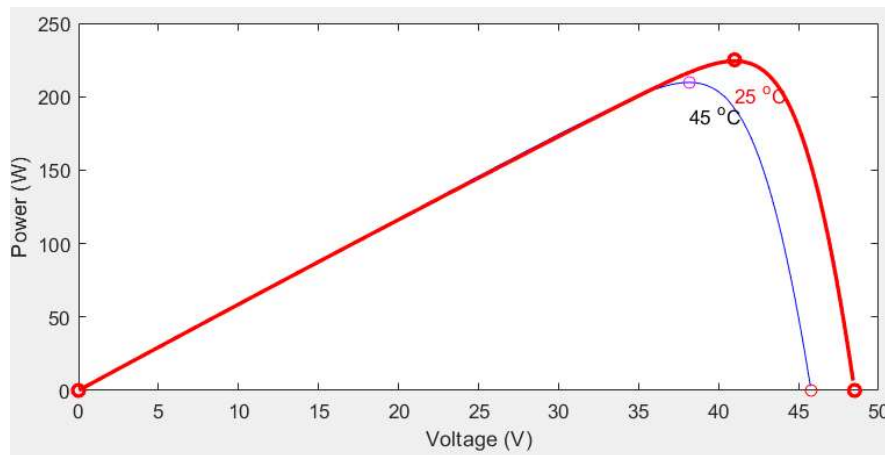


Figura 15: Efecto de la temperatura en la curva $V - P$ panel FV (a 1000 W/m^2)

Como se ha podido observar en las ilustraciones anteriores, el comportamiento del módulo fotovoltaico y sus dependencias con la irradiancia solar y la temperatura ambiente es idéntico al comportamiento de la célula solar fotovoltaica.

4.9. Fenómeno *hot spot*

Un punto caliente (*hot spot*) es una región localizada en un módulo fotovoltaico cuya temperatura de funcionamiento es muy alta en comparación con los puntos adyacentes.

Esto ocurre cuando una célula genera menos corriente que el resto de las células conectadas en serie, a causa de sombreado parcial, daño celular, desajuste o fallo de interconexión. Como resultado, la célula afectada queda polarizada al revés por medio del diodo de derivación, adquiriendo un valor de tensión igual, pero de signo contrario al resto de células, y se comporta como una carga que disipa la energía generada por el resto de las células del grupo en forma de calor, incrementando así considerablemente la temperatura en la región y degradando el rendimiento de la potencia de salida del panel fotovoltaico.

El diodo de derivación se conecta con polaridad inversa, en paralelo con un grupo de células en serie, así, la célula defectuosa queda polarizada al revés y permite la conducción hacia adelante del diodo de derivación, que provoca un cortocircuito en el grupo de células y asegura que, en el peor de los casos, la célula mencionada disipa parte de la energía generada por las células restantes en el grupo.

Dependiendo del tamaño del punto caliente resultante y de la corriente inversa, la temperatura local puede ser tan alta que puede dañar el módulo.

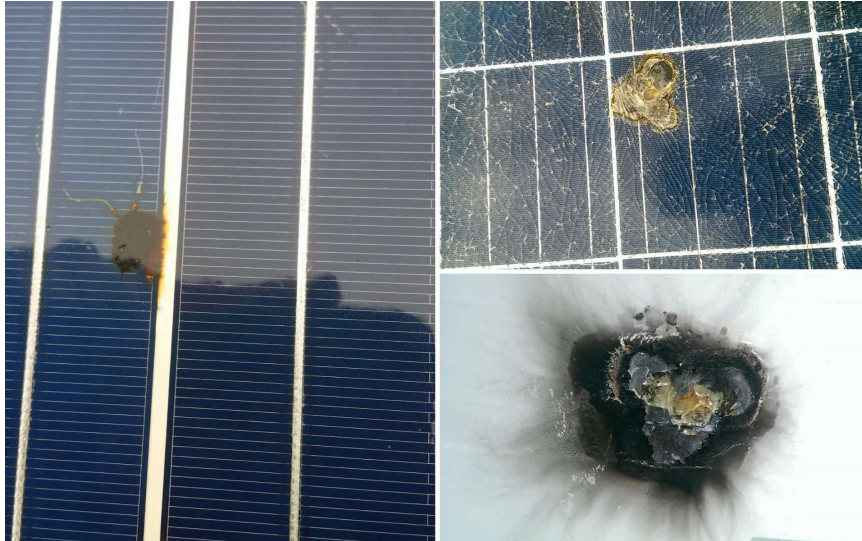


Figura 16: Daños puntos calientes

El valor que puede llegar a alcanzar esta temperatura depende principalmente de:

- Nivel de sombreado: El factor de sombreado afecta a la cantidad de irradiación solar que recibe la célula y también a la cantidad de potencia que la célula sombreada disipa, por tanto, al valor que puede alcanzar la temperatura en la célula sombreada.
- Número de células por diodo de paso: El número de células por diodo de paso determina el número de células que estarían entregando potencia a la célula sombreada, por tanto, afecta a la cantidad de potencia que disiparía la célula sombreada, influyendo así en el valor de la temperatura.

En conclusión, este tipo de defectos provocan procesos irreversibles locales como el daño en la zona del punto caliente de la celda (ruptura, reblandecimiento del material aislante), y también a nivel de sistema ya que ocasiona una disminución del rendimiento, posible destrucción del módulo completo o incluso un incendio.

5. Impacto Esperado

Con el desarrollo de este proyecto se pretende abordar uno de los problemas más temibles en la generación solar fotovoltaica. El problema es muy habitual y ocasiona grandes pérdidas económicas.

Trabajar en desarrollar un método de detección de la posible aparición de un punto caliente en su fase inicial es una cuestión emergente, pues esto facilitaría la tarea de frenar la aparición y evitar el fallo, salvando así el módulo fotovoltaico del deterioro del material que lo compone y evitando pérdidas de rendimiento. Por otra parte, se consigue evitar posibles cortocircuitos o cualquier otro incidente catastrófico como un incendio, por ejemplo.

Además de mejorar la seguridad en las instalaciones fotovoltaicas, solucionar la problemática del fenómeno de los puntos calientes se traduce en ahorros económicos y mayor cantidad de ingresos a raíz de la mejora del rendimiento de las instalaciones fotovoltaicas que supone la no aparición de puntos calientes.

Por todo esto y por la situación de fomento de la energía solar fotovoltaica que se vive actualmente a nivel nacional, podría crecer enormemente la potencia instalada en el país.

Frenar este tipo de defectos podría considerarse una forma más de fomentar la energía solar fotovoltaica.

6. Estado del arte

El método más utilizado en la actualidad para la detección de puntos calientes en paneles solares fotovoltaicos es la comprobación visual con la ayuda de una cámara termográfica. En general, los fallos en el funcionamiento de los sistemas fotovoltaicos a partir de una radiación solar de unos 600 W/m^2 se pueden diagnosticar rápidamente con una cámara termográfica con base en los cambios evidentes de las propiedades térmicas.

En algunos casos, se hace uso de drones dotados de cámaras termográficas para inspeccionar los paneles solares, lo cual resulta muy útil en tejados por ejemplo donde podría haber cientos de módulos fotovoltaicos, o en zonas de difícil acceso. Otra ventaja de usar drones con respecto al método convencional de inspección termográfica es la rapidez del procedimiento, ya que hace posible inspeccionar varios paneles solares en el mismo tiempo que requiere una inspección manual para un solo panel solar.

Por otra parte, también existen diferentes estudios complejos y profundos acerca del correcto funcionamiento de los paneles solares fotovoltaicos. Algunos parten del valor de impedancia de un determinado arreglo de celdas fotovoltaicas en serie y evalúan condiciones normales de funcionamiento en condiciones normales de operación en el punto de máxima potencia (MPP). El inconveniente de este tipo de propuestas es que el MPP ocurre para diferentes valores de corriente de máxima potencia (I_{mp}) y voltaje de máxima potencia (V_{mp}) con distintos niveles de irradiancia solar, dificultando el desarrollo de métodos para la evaluación de sombreado parcial y efectos de puntos calientes en base a la curva característica Voltaje – Corriente.

Otros estudios parten del rastreo del MPP para demostrar que ante sombreado parcial se provoca un incremento en la capacitancia y en la resistencia de un arreglo de celdas fotovoltaicas en serie, lo que se traduce en detección de puntos calientes.

Una investigación destacable propone el desarrollo de un método simple donde la impedancia DC equivalente del módulo fotovoltaico aumenta con el sombreado parcial. Conocidos los valores resistivos en condiciones normales, es posible detectar puntos calientes.

Siguiendo la misma línea, hay estudios que toman como referencia la impedancia de Thévenin (Z_{TH}) para reforzar la idea anterior, de tal manera que cuando aumenta la impedancia DC equivalente, se desconecta el panel y se mide la (Z_{TH}) para verificar el defecto.

7. Identificación y análisis de soluciones posibles

Los variables más importantes a destacar para identificar la posible aparición de un punto caliente en un panel fotovoltaico y que sufren variaciones importantes como consecuencia del fenómeno “hot spot” son la temperatura de las células fotovoltaicas, la corriente y el voltaje a la salida del panel fotovoltaico y, por ende, la potencia del mismo.

Actuar sobre la temperatura de las células solares que componen un panel fotovoltaico introduciendo sensores de temperatura para todas las células o para cada grupo de células resulta una tarea muy costosa, tanto económicamente como a nivel de labor que conlleva implementar el sistema de gestión de las mediciones y del algoritmo de diagnóstico. Sin embargo, como experimento con un único panel fotovoltaico a modo de investigación no solamente resulta factible económica y técnicamente, sino que sería muy útil para modelar este fenómeno y obtener la curva real de evolución de la temperatura bajo este fenómeno para distintas situaciones para dicho panel. Si además se combina este método con mediciones en tiempo real de corriente y de voltaje del panel fotovoltaico se puede obtener la correlación entre estas variables y la temperatura.

Al contrario que con el caso de la temperatura, actuar sobre la corriente y el voltaje de los paneles fotovoltaicos resulta más factible para implementar un sistema de detección de puntos calientes. Tan solo se necesita un sensor de corriente y un sensor de voltaje para cada panel fotovoltaico a diagnosticar en lugar de un sensor por cada célula o por cada grupo de células como es el caso de la temperatura.

Para implementar el diagnóstico en todo caso, es necesario implementar un sistema de mediciones y un sistema de diagnóstico y gestión de alarmas. Una manera de llevar esto a cabo podría ser diseñar y desarrollar los dos sistemas por separado, en distintos dispositivos y mediante un protocolo de comunicaciones converger ambos sistemas. En este caso, entra en juego también la correcta selección e implementación del protocolo de comunicaciones para garantizar una correcta y fiable comunicación entre ambos dispositivos de ambos sistemas.

Para implementar cada sistema por separado se puede recurrir a dispositivos como Raspberry pi, Beaglebone, Arduino, Odroid, autómatas, etc.

Por otra parte, también es posible implementar ambos sistemas en un mismo dispositivo, por ejemplo, un PLC dotado de módulos de expansión de entradas analógicas.

Finalmente, el mercado actual ofrece dispositivos inteligentes de medida y supervisión individuales enfocados al IoT.

8. Solución propuesta

La solución que se propone en este trabajo se basa en el análisis de las curvas de potencia en tiempo real de cada uno de los paneles fotovoltaicos a supervisar. Para ello, se diseña un sistema de medición en campo para la obtención de los valores de voltaje y corriente de cada panel fotovoltaico por separado en tiempo real. Para esta primera parte del trabajo se opta por seleccionar una tarjeta de control con un módulo de expansión de entradas analógicas, donde conectaremos los sensores de voltaje y de corriente. Las lecturas de las entradas analógicas son tratadas por el microcontrolador y transmitidas al ordenador (donde implementamos el sistema de control) vía ethernet, haciendo uso del protocolo de comunicaciones MQTT.

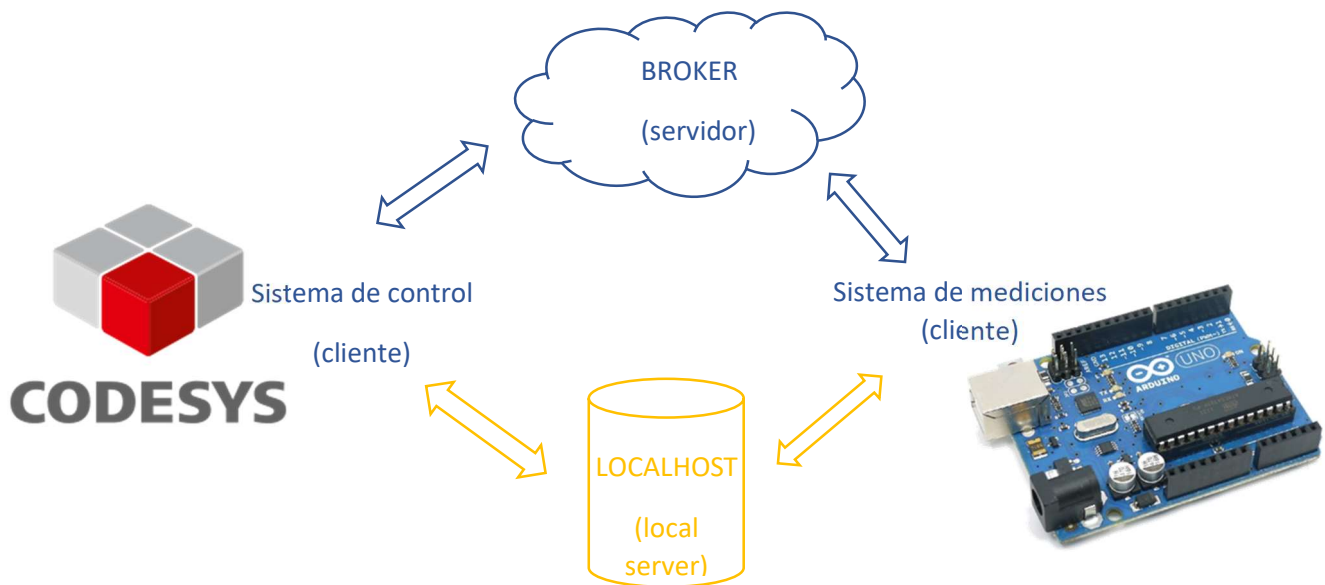


Figura 17: Esquema general solución propuesta

El siguiente paso una vez obtenidos los datos de las mediciones es construir las gráficas de potencia instantánea de cada panel fotovoltaico por separado, y de igual forma, también se construye la gráfica de la potencia media de todo el grupo de paneles (string) con los valores de todo el conjunto de paneles conectados a un mismo nodo. Esta última nos sirven de referencia, de forma que se desarrolla un algoritmo que compara en tiempo real la potencia media con las potencias de cada uno de los módulos fotovoltaicos por separado para detectar posibles desviaciones con una determinada tolerancia.

Los paneles que están otorgando una potencia por debajo del valor medio con determinado porcentaje de desviación serán sospechosos de estar bajo sombreado parcial o suciedad, y por tanto, podrían sufrir la aparición de puntos calientes.

Finalmente se gestionan alarmas y mensajes de fallo y de advertencia en función de los niveles de tolerancia predefinidos.

El proyecto ofrece también la posibilidad de actuar sobre las salidas del dispositivo de mediciones desde el sistema de monitorización y control para diversas posibles aplicaciones.

8.1. Hardware y software utilizados

- Arduino UNO

El Arduino UNO es una placa de microcontrolador de código abierto basado en el microchip ATmega328P y desarrollado por Arduino.cc. La placa está equipada con conjuntos de pines de E/S digitales y analógicas que pueden conectarse a varias placas de expansión y otros circuitos.

Se programa con el Arduino IDE (Entorno de desarrollo integrado) a través de un cable USB tipo B.3. Puede ser alimentado por el cable USB o por una batería externa de 9 voltios, aunque acepta voltajes entre 7 y 20 voltios.



Figura 18: Placa Arduino UNO

- Expansión entradas analógicas Velleman_KA12

Arduino UNO™ está equipado con 6 entradas analógicas. Sin embargo, para el presente proyectos se necesita más entradas. Este Shield utiliza sólo 4 puertos I/O (3 x digital, 1 x analógico) pero añade 24 entrada. Por ello, están disponibles 29 entradas analógicas.

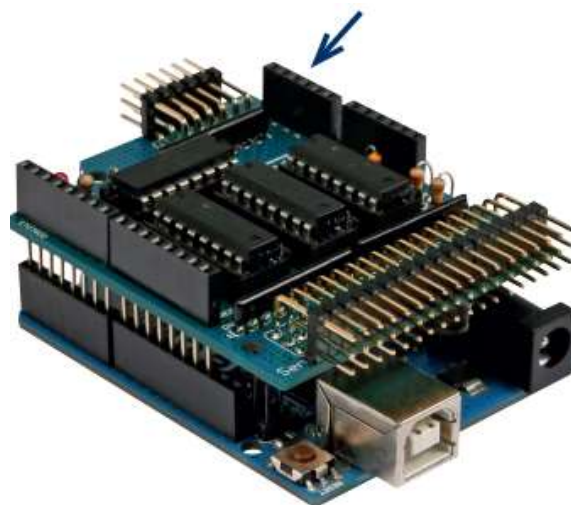


Figura 19: Expansión entradas analógicas Velleman_KA12

- Tarjeta de red Ethernet W5100

La tarjeta W5100 está diseñada para facilitar la implementación de conectividad a internet sin necesidad de un SO, lo que lo hace interesante para MCU y aplicaciones de IoT.

Incluye una pila de TCP/IP por hardware y buffer interno de 16Kbytes para Tx/Rx. Esto permite liberar de estas tareas al procesador, siendo una de sus principales ventajas frente a otros controladores de Ethernet como el ENC28J60.

El W5100 puede conectarse mediante SPI, por lo que es muy sencilla la conexión con la mayoría de los procesadores. También incorpora un lector de tarjeta SD, donde podemos guardar archivos (html, txt, jpg, png) con los que trabajar cuando actuemos como servidor.

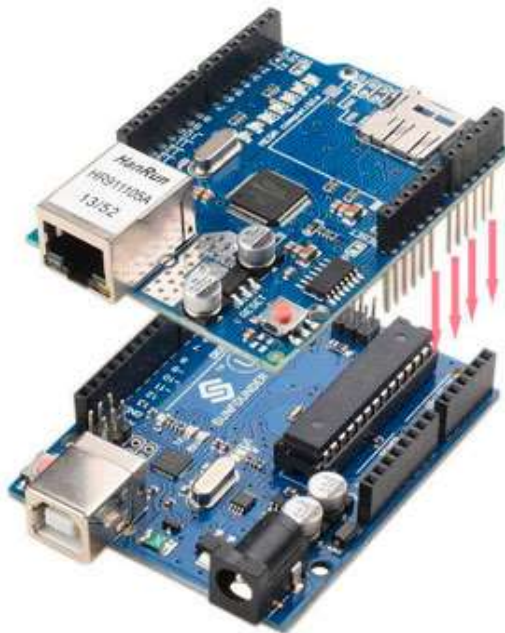


Figura 20: Tarjeta de red Ethernet W5100

- Transductor de voltaje DC

El sensor de voltaje DC de la marca Celsa transforma la entrada de voltaje (0.01 – 150 V DC) en una corriente proporcional de salida estándar (4 – 20 mA) en sus bornes de salida.

Este tipo de transductor acepta voltajes de alimentación de 20 – 250 V AC/DC.



Figura 21: Transductor de voltaje

- Transductor de corriente DC

El sensor de corriente DC de la marca Celsa transforma la entrada de corriente (0.01 – 15 A) en una corriente proporcional de salida estándar (4 – 20 mA) en sus bornes de salida.

Este tipo de transductor acepta voltajes de alimentación de 20 – 250 V AC/DC.



Figura 22: Transductor de corriente

- Arduino IDE

El entorno de desarrollo integrado (IDE) de Arduino es una aplicación multiplataforma (para Windows, macOS, Linux) que está escrita en el lenguaje de programación Java. Se utiliza para escribir y cargar programas en placas compatibles con Arduino principalmente. El código fuente para el IDE se publica bajo la Licencia Pública General de GNU.



Figura 23: Arduino IDE (entorno de desarrollo integrado)

El IDE de Arduino admite los lenguajes C y C++ utilizando reglas especiales de estructuración de códigos y suministra una biblioteca de software del proyecto Wiring, que proporciona muchos procedimientos comunes de E/S.

El código escrito por el usuario solo requiere dos funciones básicas, para iniciar el boceto y el ciclo principal del programa, que se compilan y vinculan con un apéndice de programa main() en un ciclo con el GNU toolchain, que también se incluye. El IDE de Arduino emplea el programa avrdude para convertir el código ejecutable en un archivo de texto en codificación hexadecimal que se carga en la placa Arduino mediante un programa de carga en el firmware de la placa.

- CODESYS

CODESYS es un entorno de desarrollo para la programación de controladores conforme con el estándar industrial internacional IEC 61131-3. El término CODESYS es un acrónimo y significa Sistema de Desarrollo de Controladores.

Más de 250 fabricantes de dispositivos de diferentes sectores industriales ofrecen sus dispositivos de automatización inteligente programable con la interfaz de programación CODESYS. En consecuencia, miles de usuarios finales en todo el mundo emplean CODESYS para su trabajo diario en todo tipo de tareas de automatización. Hoy en día, CODESYS es la herramienta de desarrollo basada en IEC 61131-3 más extendida en Europa.

Una red mundial de asociados del sistema de CODESYS ofrece tanto una amplia variedad de servicios para los usuarios CODESYS como el apoyo a los usuarios finales, soporte, consultoría, formación, programación de aplicaciones o la integración de sistemas.

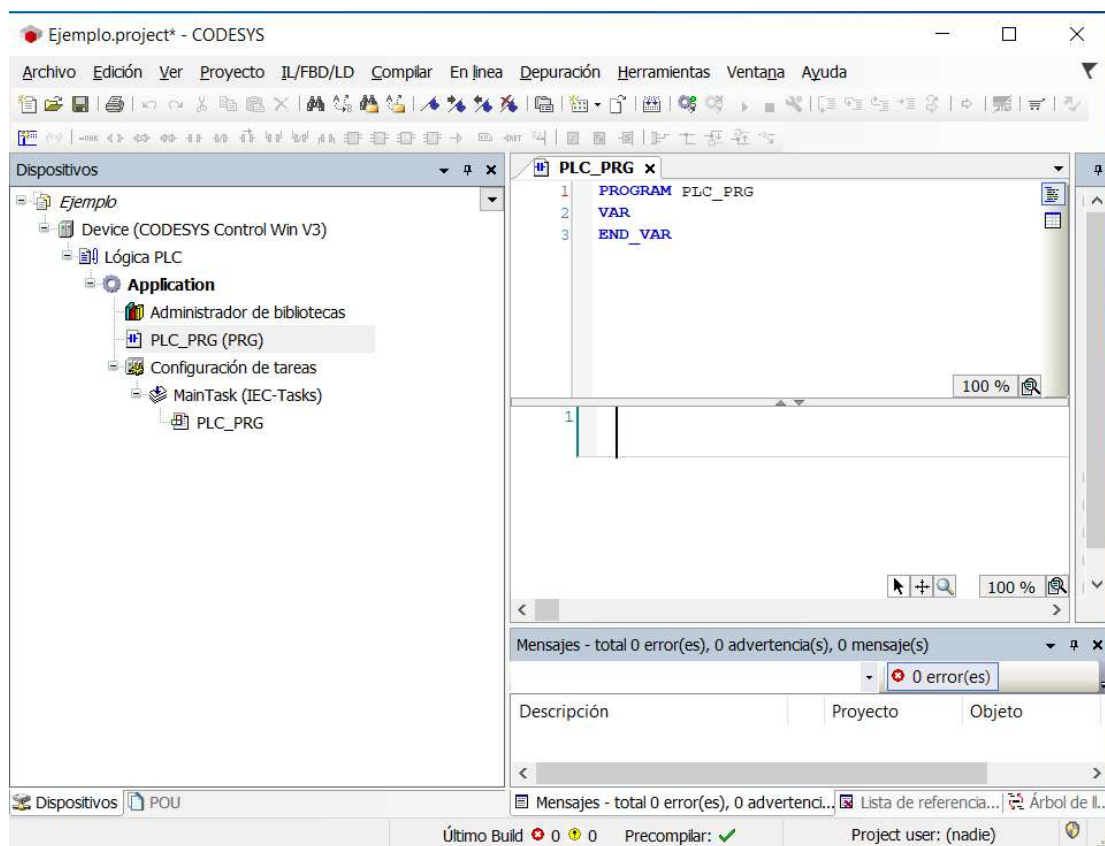


Figura 24: Entorno de desarrollo Codesys

CodeSys utiliza el estándar de programación de PLC IEC 61131-3 con 5 lenguajes de programación:

- Lenguaje escalera (LD - Ladder Diagram)
- Diagrama de bloque de funciones (FBD - Function Block Diagram)
- Texto estructurado (ST - Structured Text)
- Lista de instrucciones (IL - Instruction List)
- Bloques de función secuenciales (SFC - Sequential Function Chart)

- CODESYS Control Win SL

El CODESYS Control Win SL convierte un PC industrial en un PLC de alto rendimiento de uso universal - casi arbitrariamente escalable a través del rendimiento del PC. El sistema es adecuado para aplicaciones sin requisitos exigentes sobre el comportamiento en tiempo real.

El sistema de tiempo de ejecución soporta numerosas interfaces de E/S, como por ejemplo tarjetas de entrada/salida discretas o de bus de campo, así como pilas de protocolo IEC 61131-3 integradas. Los buses de campo se configuran directamente en el Sistema de Desarrollo CODESYS - sin el uso de herramientas adicionales.



Figura 25: Software CODESYS Control Win SL

- CODESYS SoftMotion

CODESYS SoftMotion es una opción adicional para los sistemas SoftPLC compatibles con CODESYS. CODESYS SoftMotion amplía el alcance funcional de estos sistemas de un controlador puramente lógico a un controlador de movimiento, opcionalmente también con soporte CNC y robótico.



Figura 26: Software CODESYS SoftMotion

- Codesys Gateway

Codesys Gateway conecta el CODESYS Automation Server a los PLCs de CODESYS en una red local. El paquete está disponible en versión Windows y Linux. permite la comunicación entre el servidor de automatización de CODESYS y los controladores conectados. Puede funcionar en un controlador o en un dispositivo autónomo de la red local.

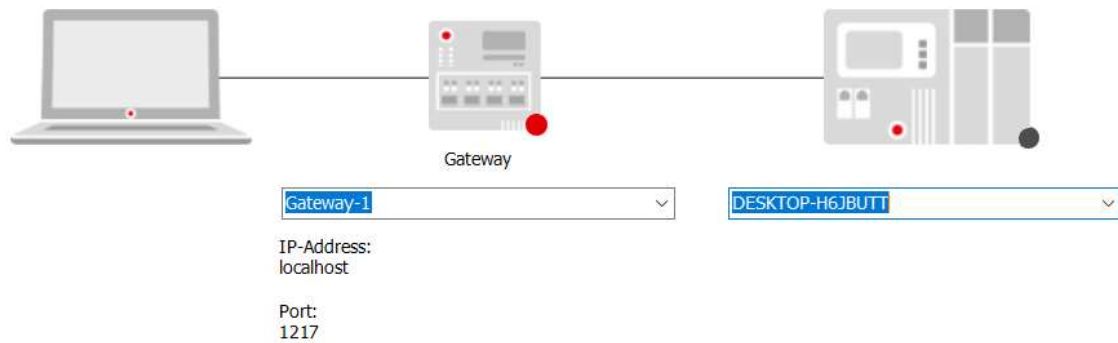


Figura 27: CODESYS Gateway

- Broker shiftr.io

Shiftr.io es un Broker gratuito enfocado a aplicaciones IoT que ofrece la posibilidad de trabajar en la nube o instalarse el Broker en el PC ofreciendo así también la posibilidad de trabajar con un servidor local ubicado en la propia computadora.

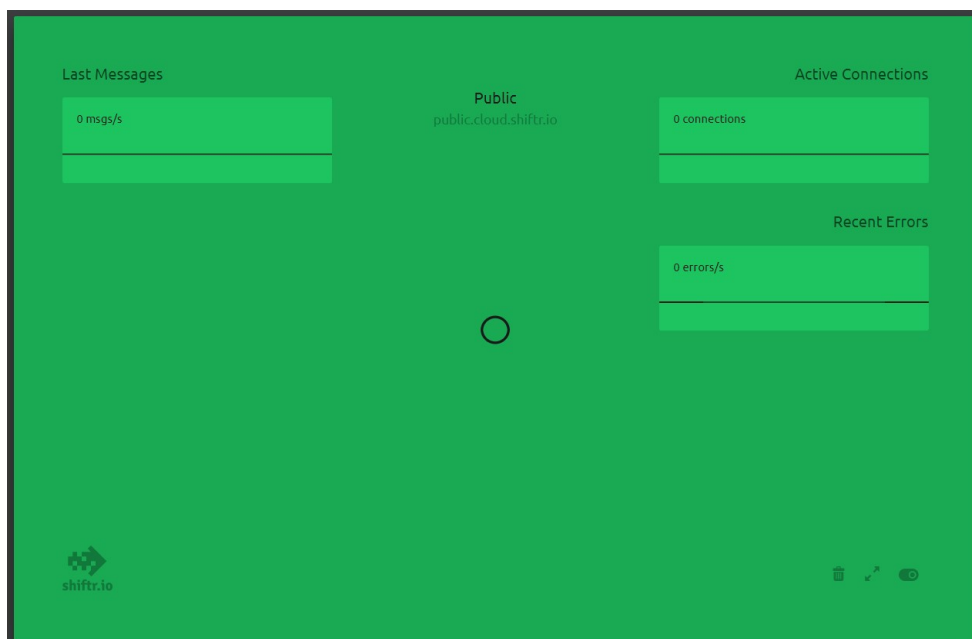


Figura 28: Broker shiftr.io en la nube

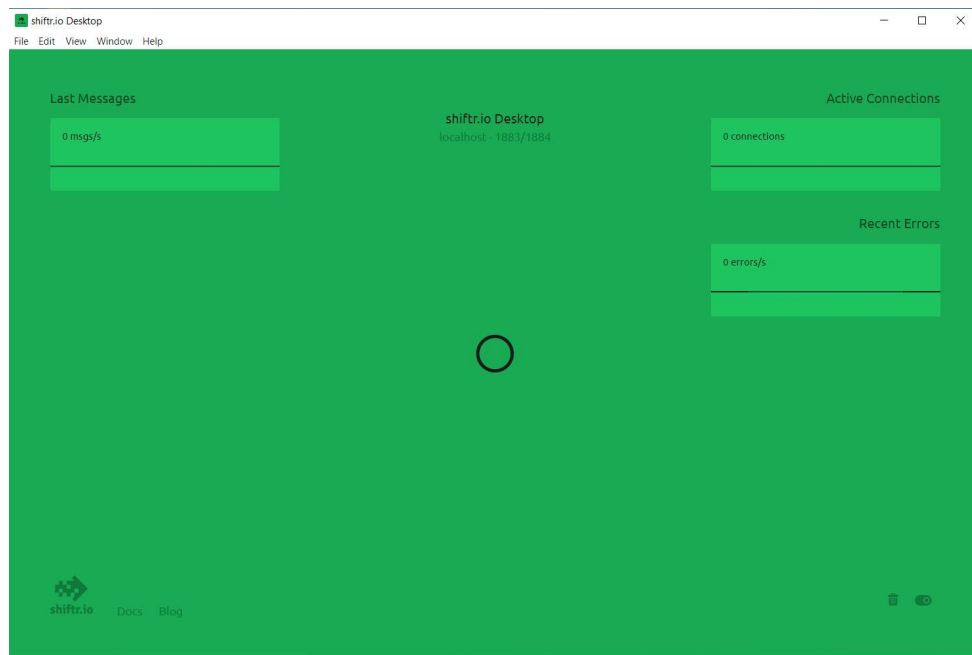


Figura 29: Broker shiftr.io en el PC (aplicación de escritorio)

8.2. Diseño de la solución propuesta

La solución propuesta consta de tres partes importantes: diseñar un sistema de mediciones en campo para recoger las medidas en tiempo real voltaje y corriente de los paneles solares para posteriormente tratar los datos, escoger un sistema de control para la monitorización y gestión de alarmas, aquí entra la recepción de los datos y su posterior tratamiento, la visualización en tiempo real, el desarrollo del algoritmo para la detección de posibles puntos calientes y la programación de las notificaciones y alarmas así como la gestión de las mismas y finalmente escoger e implementar un protocolo de comunicaciones apropiado.

El sistema de mediciones está compuesto por un dispositivo Arduino UNO, un módulo de expansión de entradas analógicas (29 entradas analógicas en total por cada Arduino) y una tarjeta de red Ethernet, transductores de corriente con salida 4 – 20 mA y transductores de voltaje con salida también de 4 – 20 mA.

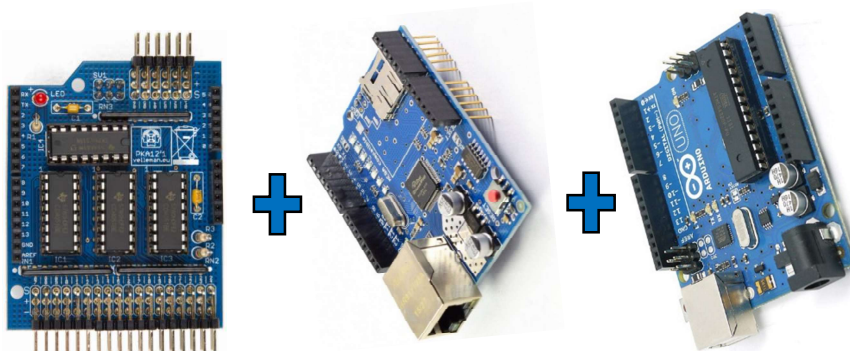


Figura 30: Sistema de mediciones



Figura 31: Transductores de corriente y de voltaje

El sistema de monitorización y gestión de alarmas para este trabajo se concentra en el PC, que aprovecha la ventaja que ofrece Codesys de PLC virtual, no obstante, es perfectamente factible emplear un PLC real junto con una interfaz HMI para la monitorización o junto con un PC. Para el caso de este proyecto, nuestro PC tomará el rol de PLC y de interfaz de monitorización gracias a las herramientas que ofrece Codesys.

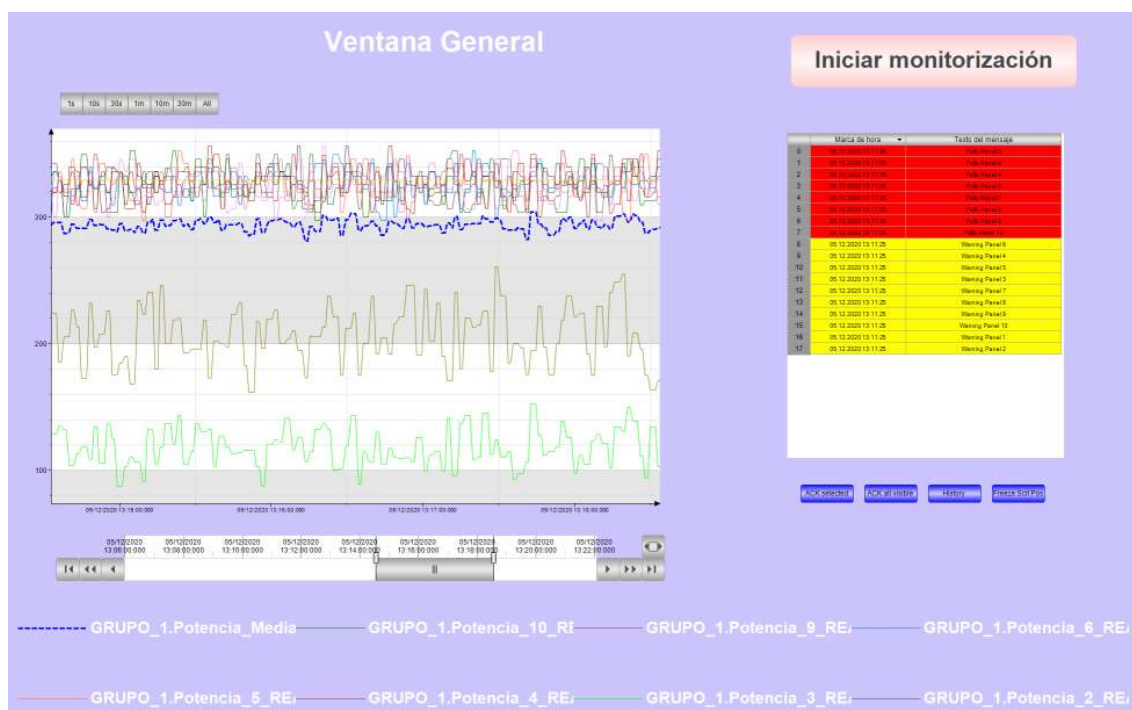


Figura 32: Ventana general de monitorización

En lo que respecta al protocolo de comunicaciones escogido es MQTT (Message Queue Telemetry Transport), un protocolo usado para la comunicación "machine-to-machine" (M2M) en el "Internet of Things". Este protocolo está orientado a la comunicación de sensores, debido a que consume muy poco ancho de banda y puede ser utilizado en la mayoría de los dispositivos empotrados con pocos recursos.

La arquitectura de MQTT sigue una topología de estrella, con un nodo central que hace de servidor o broker al que pueden suscribirse un número determinado de cliente (generalmente muy alto, decenas de miles). La comunicación se basa en topics (temas), que el cliente que publica el mensaje crea y los nodos que deseen recibirlo deben suscribirse al topic que les interesa. La comunicación puede ser de uno a uno, o de uno a muchos. Un topic se representa mediante una cadena de caracteres (string).

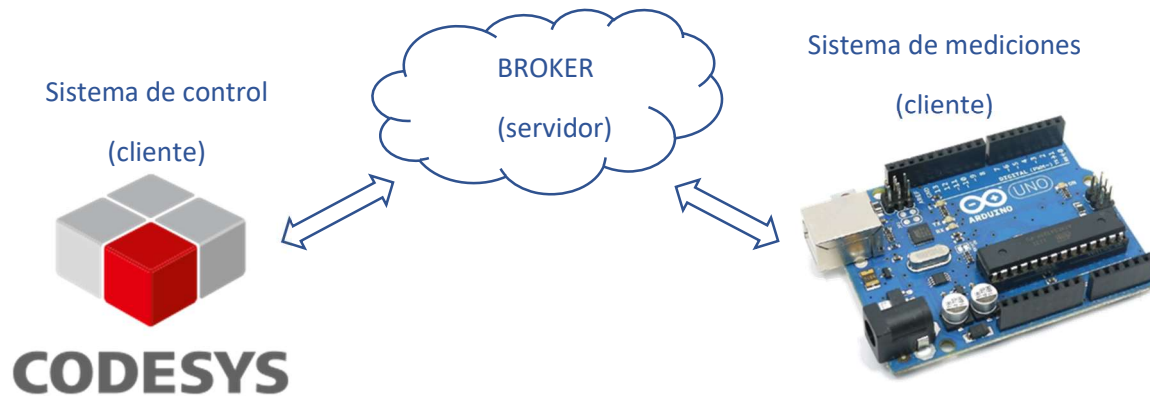


Figura 33: Esquema general sistemas de control y de mediciones con broker en la nube

8.3. Desarrollo de la solución propuesta

Haciendo uso de los transductores descritos en el apartado anterior, y las entradas analógicas del sistema de mediciones tomamos lecturas de valores de voltaje y de corriente. Los transductores emiten señales de corriente a las entradas analógicas de 4 – 20 mA en proporción al valor de los valores reales. Arduino interpreta estas señales como un número entero entre 0 y 1023, también proporcional a la señal, de modo que es necesario transformar este número digital en un valor de voltaje o de corriente, según corresponda, por código. También por código, se calcula la potencia que genera cada panel a partir de las mediciones recogidas de voltaje y corriente de cada panel.

El sistema de mediciones se encarga también, mediante la tarjeta de red Ethernet, de establecer comunicación como cliente con un broker previamente definido en la red o instalado en el propio computador, y enviar los datos recogidos y tratados al broker en diferentes topics. Concretamente se envían los datos de potencia, cada dato se publica en el broker bajo un topic acorde a la numeración del panel al que pertenece.

```
Arduino_MQTT_V2
//----- Leer sensores panel 1, calcular potencia y enviar resultado al brocker -----//
float A1_Voltaje_1 = analogRead(A0) / 1023.0 * MAX_VOLTAGE;
float A1_Corriente_1 = analogRead(A1) / 1023.0 * MAX_CURRENT;
float A1_Potencia_1 = A1_Voltaje_1 * A1_Corriente_1;

client.publish(topic_pub_Arduino1_Panel_1, String(A1_Potencia_1), true, 1);

//----- Leer sensores panel 2, calcular potencia y enviar resultado al brocker -----//
float A1_Voltaje_2 = A1_Voltaje_1 * random(95,105)/100.0;
float A1_Corriente_2 = A1_Corriente_1 * random(95,105)/100.0;
float A1_Potencia_2 = A1_Voltaje_2 * A1_Corriente_2;

client.publish(topic_pub_Arduino1_Panel_2, String(A1_Potencia_2), true, 1);

//----- Leer sensores panel 3, calcular potencia y enviar resultado al brocker -----//
float A1_Voltaje_3 = A1_Voltaje_1 * random(50,70)/100.0;
float A1_Corriente_3 = A1_Corriente_1 * random(50,70)/100.0;
float A1_Potencia_3 = A1_Voltaje_3 * A1_Corriente_3;

client.publish(topic_pub_Arduino1_Panel_3, String(A1_Potencia_3), true, 1);

//----- Leer sensores panel 4, calcular potencia y enviar resultado al brocker -----//
float A1_Voltaje_4 = A1_Voltaje_1 * random(95,105)/100.0;
float A1_Corriente_4 = A1_Corriente_1 * random(95,105)/100.0;
float A1_Potencia_4 = A1_Voltaje_4 * A1_Corriente_4;

client.publish(topic_pub_Arduino1_Panel_4, String(A1_Potencia_4), true, 1);

//----- Leer sensores panel 5, calcular potencia y enviar resultado al brocker -----//
float A1_Voltaje_5 = A1_Voltaje_1 * random(95,105)/100.0;
float A1_Corriente_5 = A1_Corriente_1 * random(95,105)/100.0;
float A1_Potencia_5 = A1_Voltaje_5 * A1_Corriente_5;

client.publish(topic_pub_Arduino1_Panel_5, String(A1_Potencia_5), true, 1);
```

Figura 34: Lectura de voltaje y de corriente, cálculo de la potencia y publicación de ésta en el broker.

Para implementar el protocolo de comunicaciones MQTT, existen varios servidores (brokers) gratuitos. Para el caso de este trabajo se hace uso de la plataforma shiftr.io que ofrece un servidor gratuito tanto en la nube como local.

El broker shiftr.io es donde publica el sistema de mediciones los datos de potencia de cada panel en sus respectivos topics, de manera que es el enlace entre el mencionado sistema de mediciones y el sistema de control.

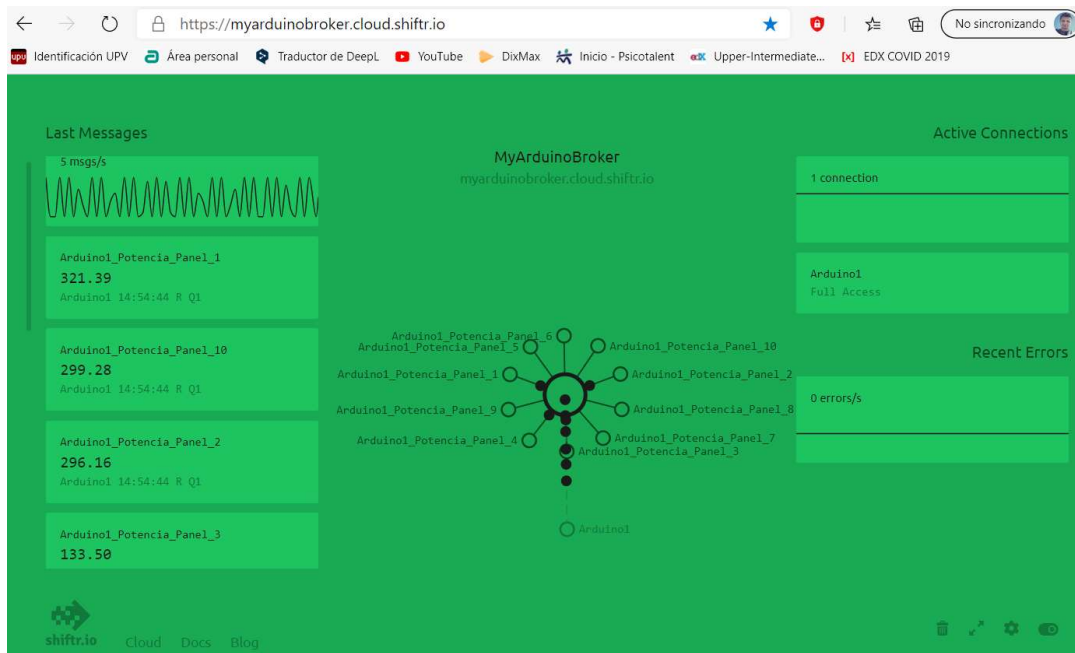


Figura 35: Sistema de mediciones publicando las potencias de los paneles en el broker shiftr.io

Finalmente, el sistema de control establece comunicación con el broker como cliente y se suscribe a los topics que publica el sistema de mediciones para así recibir los mensajes de dichos topics y tratarlos apropiadamente. Así como ofrecer un entorno de monitorización, supervisión y gestión de alarmas. Las alarmas se definen en base al algoritmo de detección de anomalías establecido.

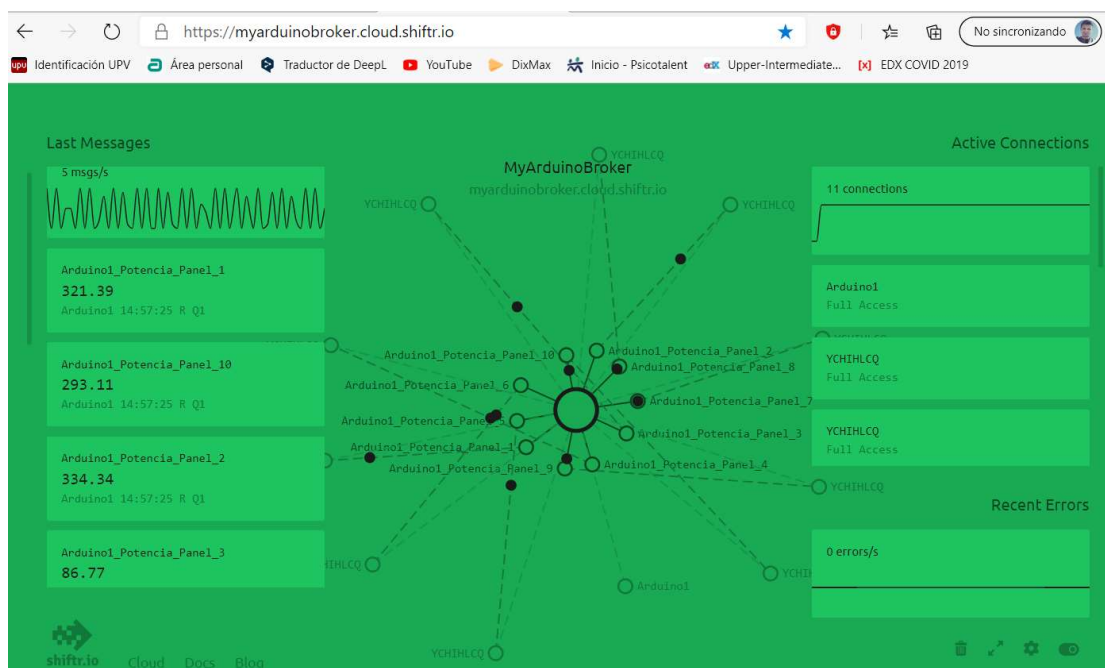


Figura 36: Sistema de control suscrito a los topics donde publica el sistema de mediciones y recibiendo los mensajes de dichos topics

El algoritmo establecido para la detección de anomalía en los paneles solares y por ende posible fase inicial de aparición de punto caliente consiste en la obtención del valor medio de las potencias que genera cada panel en tiempo real y simultáneamente comparar este valor medio con los valores individuales de cada panel fotovoltaico conectado al sistema de mediciones fuente de los datos, fijando unos valores de desviación estándar, en caso de que algún panel resulte otorgar una potencia inferior a la potencia media – la tolerancia definida, será sospechoso y deberá revisarse.

8.4. Implantación

Por cada panel fotovoltaico se necesita un sensor de voltaje, un sensor de corriente y dos entradas analógicas. Además, para cada sistema de mediciones (conjunto de paneles) se asigna una variable de salida de reserva para aplicaciones futuras, de modo que desde el sistema de control es posible activar y desactivar dicha variable booleana que a su vez actuará directamente sobre la salida que tenga asignada en el sistema de mediciones (pin conexión Arduino).

A continuación, la tabla de distribución de E/S del sistema de mediciones:

Tabla pines de conexión	
Elemento	Pin de conexión Arduino UNO
Sensor de Voltaje Panel 1	A0 (entrada analógica 0)
Sensor de Corriente Panel 1	A1 (entrada analógica 1)
Sensor de Voltaje Panel 2	A2 (entrada analógica 2)
Sensor de Corriente Panel 2	A3 (entrada analógica 3)
Sensor de Voltaje Panel 3	A4 (entrada analógica 4)
Sensor de Corriente Panel 3	A5 (entrada analógica 5)
Sensor de Voltaje Panel 4	A6 (entrada analógica 6)
Sensor de Corriente Panel 4	A7 (entrada analógica 7)
Sensor de Voltaje Panel 5	A8 (entrada analógica 8)
Sensor de Corriente Panel 5	A9 (entrada analógica 9)
Sensor de Voltaje Panel 6	A10 (entrada analógica 10)
Sensor de Corriente Panel 6	A11 (entrada analógica 11)
Sensor de Voltaje Panel 7	A12 (entrada analógica 12)
Sensor de Corriente Panel 7	A13 (entrada analógica 13)
Sensor de Voltaje Panel 8	A14 (entrada analógica 14)
Sensor de Corriente Panel 8	A15 (entrada analógica 15)
Sensor de Voltaje Panel 9	A16 (entrada analógica 16)
Sensor de Corriente Panel 9	A17 (entrada analógica 17)
Sensor de Voltaje Panel 10	A18 (entrada analógica 18)
Sensor de Corriente Panel 10	A19 (entrada analógica 19)
Variable de salida 1 (VAR_OUT_1)	7 (salida digital 7)

Figura 37: Tabla pines de conexión de los sensores

Para el caso de este trabajo académico, se utiliza 2 potenciómetros que simulan los sensores de voltaje y de corriente conectados en las entradas analógicas A0 y A1 respectivamente (panel 1). Mientras que los valores de los demás sensores (panel 2 a panel 10) son simulados con la

generación de valores aleatorios con desviaciones predefinidas con respecto a los valores de las entradas A0 y A1 (panel 1).

Las señales de los transductores de 4 – 20 mA que percibe Arduino como un número entero entre 0 y 1023 se transforman a valores de tensión y de corriente (según corresponda) de manera sencilla conociendo los rangos de medición de los sensores:

```
#define MAX_VOLTAGE 150.0
#define MAX_CURRENT 15.0 ,

//----- Leer sensores panel 1, calcular potencia y enviar resultado al broker -----//
float A1_Voltaje_1 = analogRead(A0) / 1023.0 * MAX_VOLTAGE;
float A1_Corriente_1 = analogRead(A1) / 1023.0 * MAX_CURRENT;
float A1_Potencia_1 = A1_Voltaje_1 * A1_Corriente_1;

client.publish(topic_pub_Arduino1_Panel_1, String(A1_Potencia_1), true, 1);
```

Figura 38: Código Arduino lectura sensores, conversión valores, cálculo potencia y publicación en el broker

Una vez obtenidos los valores reales de voltaje y corriente, se obtiene la potencia como resultado del producto de ambas variables y se publica dicho valor en el broker en su correspondiente topic. Para publicar cualquier mensaje en el broker es necesario que tenga un formato de cadena de caracteres (string), por este motivo se convierte el valor de la potencia en cadena de caracteres (string) justo antes de publicar.

Esta operación de publicar es posible gracias a la librería “MQTTClient.h” para Arduino que contiene las funciones necesarias para implementar el protocolo MQTT haciendo uso del dispositivo Arduino UNO. Y por supuesto, también gracias a la librería “Ethernet.h” que permite utilizar la tarjeta de red del sistema de mediciones correctamente y tener acceso a Internet.

```
#include <Ethernet.h>
#include <MQTTClient.h>

EthernetClient net;
MQTTClient client;
```

Figura 39: Librerías necesarias Arduino

Un punto importante para lograr una correcta comunicación con el broker es la correcta configuración de los datos de entrada propios del broker a la librería:

```
//----- Broker MQTT -----//
const char* broker = "myarduinobroker.cloud.shiftr.io";
char mqttUserName[] = "myarduinobroker";
char mqttPass[] = "9876543210";
```

Figura 40: Configuración broker en Arduino

Para poder publicar en un topic en el broker se tiene que definir previamente como const char* y se le asigna un nombre a la variable a publicar. De igual forma, para suscribir el sistema de mediciones a un topic del broker, se crea una variable const char* y se le asigna nombre de topic.

```
//----- Definir TOPICS -----//

//----- Topics para publicar las potencias de los paneles del Arduino n°1
const char* topic_pub_Arduino1_Panel_1 = "Arduino1_Potencia_Panel_1"; //
const char* topic_pub_Arduino1_Panel_2 = "Arduino1_Potencia_Panel_2"; //
const char* topic_pub_Arduino1_Panel_3 = "Arduino1_Potencia_Panel_3"; //
const char* topic_pub_Arduino1_Panel_4 = "Arduino1_Potencia_Panel_4"; //
const char* topic_pub_Arduino1_Panel_5 = "Arduino1_Potencia_Panel_5"; //
const char* topic_pub_Arduino1_Panel_6 = "Arduino1_Potencia_Panel_6"; //
const char* topic_pub_Arduino1_Panel_7 = "Arduino1_Potencia_Panel_7"; //
const char* topic_pub_Arduino1_Panel_8 = "Arduino1_Potencia_Panel_8"; //
const char* topic_pub_Arduino1_Panel_9 = "Arduino1_Potencia_Panel_9"; //
const char* topic_pub_Arduino1_Panel_10 = "Arduino1_Potencia_Panel_10"; /

// Topic para leer publicaciones de Codesys.
const char* topic_sub_CoDeSys = "CoDeSys";
```

Figura 41: Código Arduino para declaración topics

En cuanto al sistema de control, se realiza un programa en Codesys haciendo uso de la biblioteca MQTT Client Library para establecer la comunicación con el broker shiftr.io en la nube donde publica el sistema de mediciones.

Para cada bloque se crea una variable de tipo MQTT_Client.FB_MQTTClient que nos sirve para ponerle un nombre al bloque acorde a la aplicación que le vamos a dar, una variable de tipo cadena de caracteres (string) para almacenar el dato leído desde el broker y una variable de tipo flotante (real) para almacenar el dato leído del broker tras convertirlo a real.

```
VAR
// Panel fotovoltaico 1
Arduino1_Panel_1: MQTT_Client.FB_MQTTClient;
Potencial: STRING; // Última lectura (STRING)
Potencia_1_REAL:REAL; // Última lectura W (REAL)
```

Figura 42: Variables a definir por cada panel en Codesys

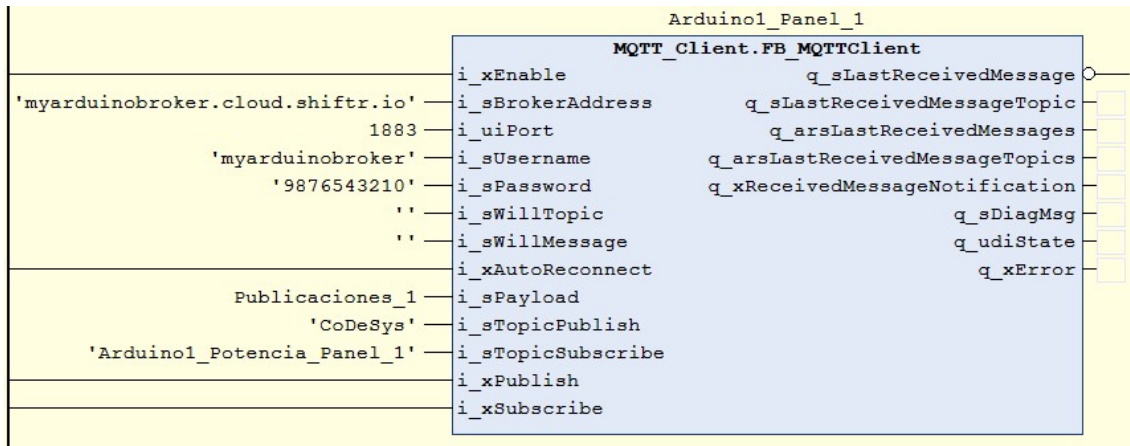


Figura 43: Bloque Cliente MQTT Codesys

El bloque anterior es el responsable de establecer la comunicación con el broker shiftr.io. Se implementa un bloque por cada panel, de modo que cada bloque recoge el dato publicado en un topic distinto en función del panel al que pertenece.

Además, para activar y desactivar la salida de reserva del sistema de mediciones, se declara una variable de entrada (VAR_INPUT) de tipo string para publicar el estado de la variable booleana que se crea en el sistema de visualización para conmutar dicha salida, de modo que desde el sistema de monitorización podemos poner a 1 o a 0 la salida de reserva.

Por otra parte, se declara una variable más de entrada en la que se registra la potencia media para cada periodo de muestreo que se calcula desde un programa auxiliar a partir de las lecturas realizadas. Esta potencia media registrada es la referencia que se toma para implementar el algoritmo de detección de anomalías en la generación de los paneles fotovoltaicos.

```

VAR_INPUT

// Potencia media grupo paneles 1
Potencia_Media_1:REAL;
// Variable para publicar (STRING)
Publicaciones_1: STRING;

```

Figura 44: Declaración variables de entrada

El sistema de monitorización es de un diseño sencillo, y consta de una ventana general de monitorización y una ventana individual de monitorización por cada panel. Se ha definido una variable asociada a un botón del sistema de visualización cuya función es iniciar la monitorización, y detenerla si se vuelve a pulsar dicho botón ubicado en la parte superior derecha de cada ventana.

En la ventana general se muestra en tiempo real las generaciones individuales de todos los paneles fotovoltaicos conectados al sistema de mediciones junto con la generación media en una misma gráfica con su correspondiente leyenda. La gráfica está dotada de herramientas de navegación en la línea del tiempo de monitorización, de modo que en cualquier momento se puede mostrar registros de mediciones pasadas para visualizar mejor la tendencia de las variables.

Además, también existe una tabla de gestión de alarmas, donde se notifican las alarmas previamente programadas acorde al algoritmo de detección implementado.

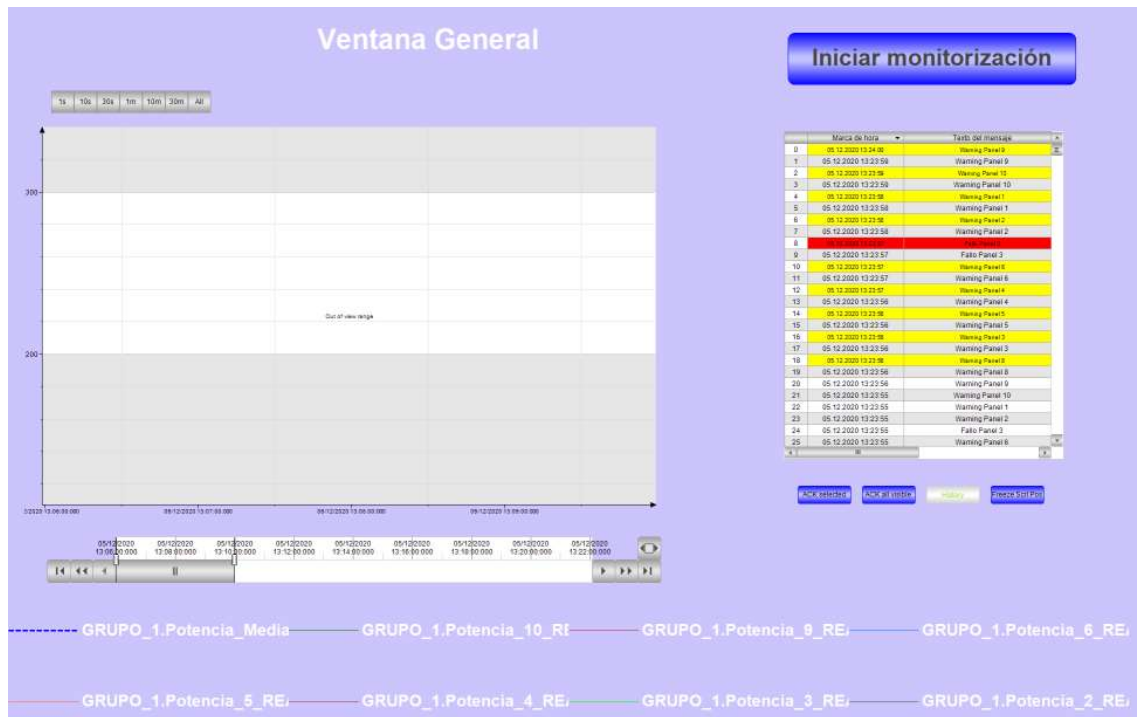


Figura 45: Ventana general de monitorización en estado reposo

Si conectamos los sistemas de control y de mediciones al broker y pulsamos en “Iniciar monitorización”, pondremos en marcha el sistema y podremos visualizar en tiempo real la generación de cada panel, la generación media y la tabla de alarmas activas.

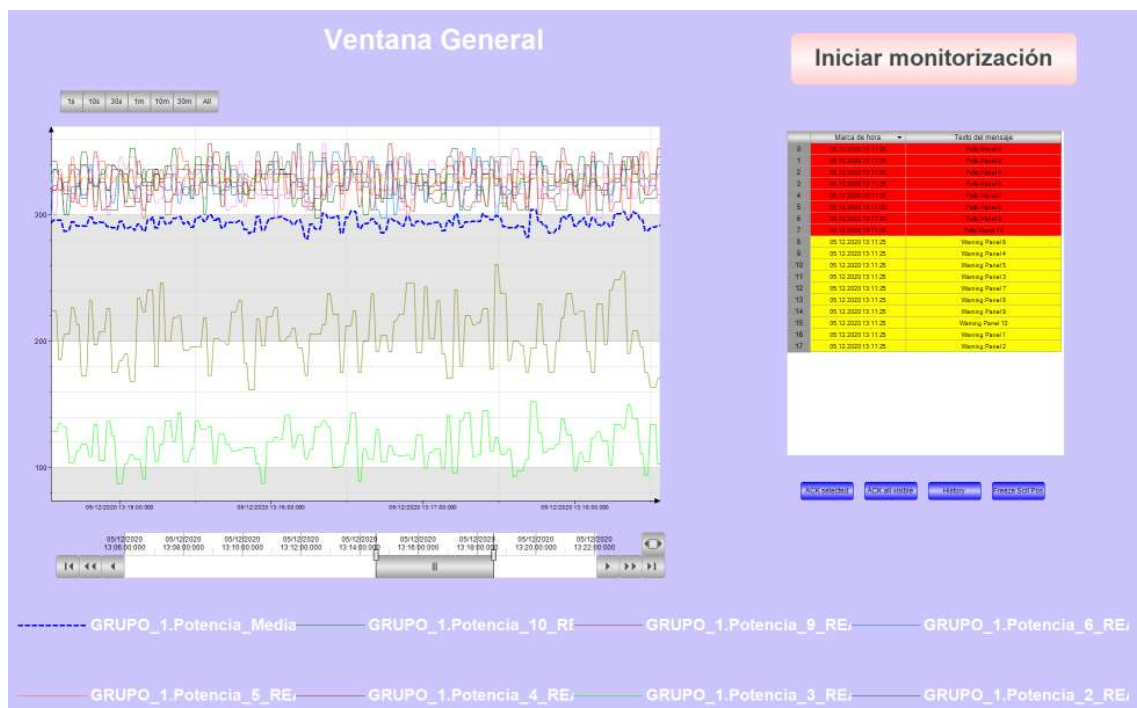


Figura 46: Ventana general de monitorización en estado activo

Desde la parte inferior de la ventana general de monitorización podemos acceder a las ventanas individuales de todos los paneles y también activar y desactivar la salida de reserva.



Figura 47: Ventana general de monitorización zona inferior

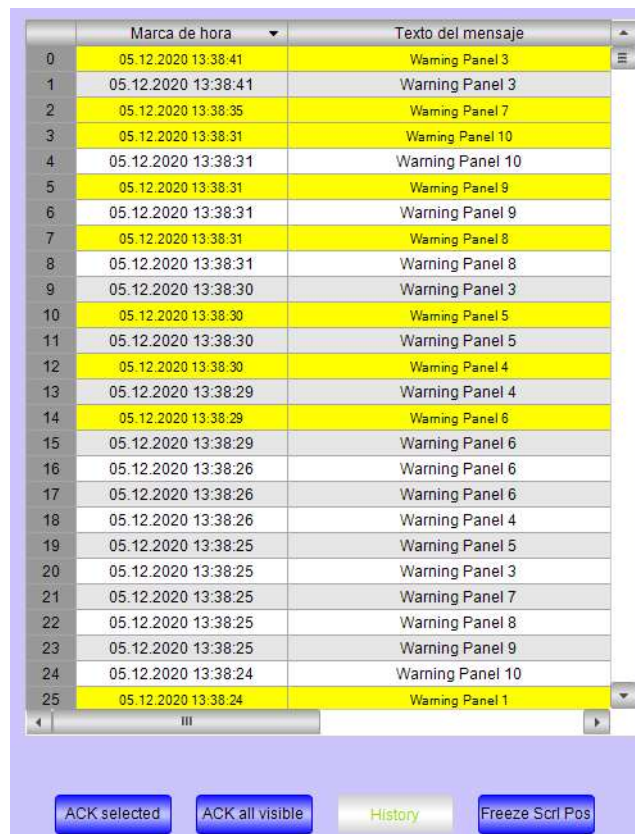
La tabla de alarmas tiene asociados unos botones definidos para algunas funciones consideradas útiles para la gestión de la tabla.

	Marca de hora	Texto del mensaje
0	05.12.2020 13:38:41	Warning Panel 3
1	05.12.2020 13:38:35	Warning Panel 7
2	05.12.2020 13:38:31	Warning Panel 10
3	05.12.2020 13:38:31	Warning Panel 9
4	05.12.2020 13:38:31	Warning Panel 8
5	05.12.2020 13:38:30	Warning Panel 5
6	05.12.2020 13:38:30	Warning Panel 4
7	05.12.2020 13:38:29	Warning Panel 6
8	05.12.2020 13:38:24	Warning Panel 1
9	05.12.2020 13:38:24	Warning Panel 2
10	05.12.2020 13:38:24	Failo Panel 3

Figura 48: Gestor de alarmas: Alarmas activas

- ACK selected: Confirmar selección
- ACK all visible: Confirmar todas las visibles
- History: Mostrar historial
- Freeze Scrl Pos: Congelar tabla

A continuación, un ejemplo del uso de la opción "History":



	Marca de hora	Texto del mensaje
0	05.12.2020 13:38:41	Warning Panel 3
1	05.12.2020 13:38:41	Warning Panel 3
2	05.12.2020 13:38:35	Warning Panel 7
3	05.12.2020 13:38:31	Warning Panel 10
4	05.12.2020 13:38:31	Warning Panel 10
5	05.12.2020 13:38:31	Warning Panel 9
6	05.12.2020 13:38:31	Warning Panel 9
7	05.12.2020 13:38:31	Warning Panel 8
8	05.12.2020 13:38:31	Warning Panel 8
9	05.12.2020 13:38:30	Warning Panel 3
10	05.12.2020 13:38:30	Warning Panel 5
11	05.12.2020 13:38:30	Warning Panel 5
12	05.12.2020 13:38:30	Warning Panel 4
13	05.12.2020 13:38:29	Warning Panel 4
14	05.12.2020 13:38:29	Warning Panel 6
15	05.12.2020 13:38:29	Warning Panel 6
16	05.12.2020 13:38:26	Warning Panel 6
17	05.12.2020 13:38:26	Warning Panel 6
18	05.12.2020 13:38:26	Warning Panel 4
19	05.12.2020 13:38:25	Warning Panel 5
20	05.12.2020 13:38:25	Warning Panel 3
21	05.12.2020 13:38:25	Warning Panel 7
22	05.12.2020 13:38:25	Warning Panel 8
23	05.12.2020 13:38:25	Warning Panel 9
24	05.12.2020 13:38:24	Warning Panel 10
25	05.12.2020 13:38:24	Warning Panel 1

Buttons: ACK selected, ACK all visible, History, Freeze Scr Pos

Figura 49: Gestor de alarmas: Historial alarmas

En cada una de las ventanas de monitorización individual podemos observar en tiempo real la evolución de la generación del panel en cuestión junto con la generación media del grupo. Todas las gráficas disponen de herramientas de navegación en el tiempo para visualizar periodos concretos y para navegar por la línea del tiempo.

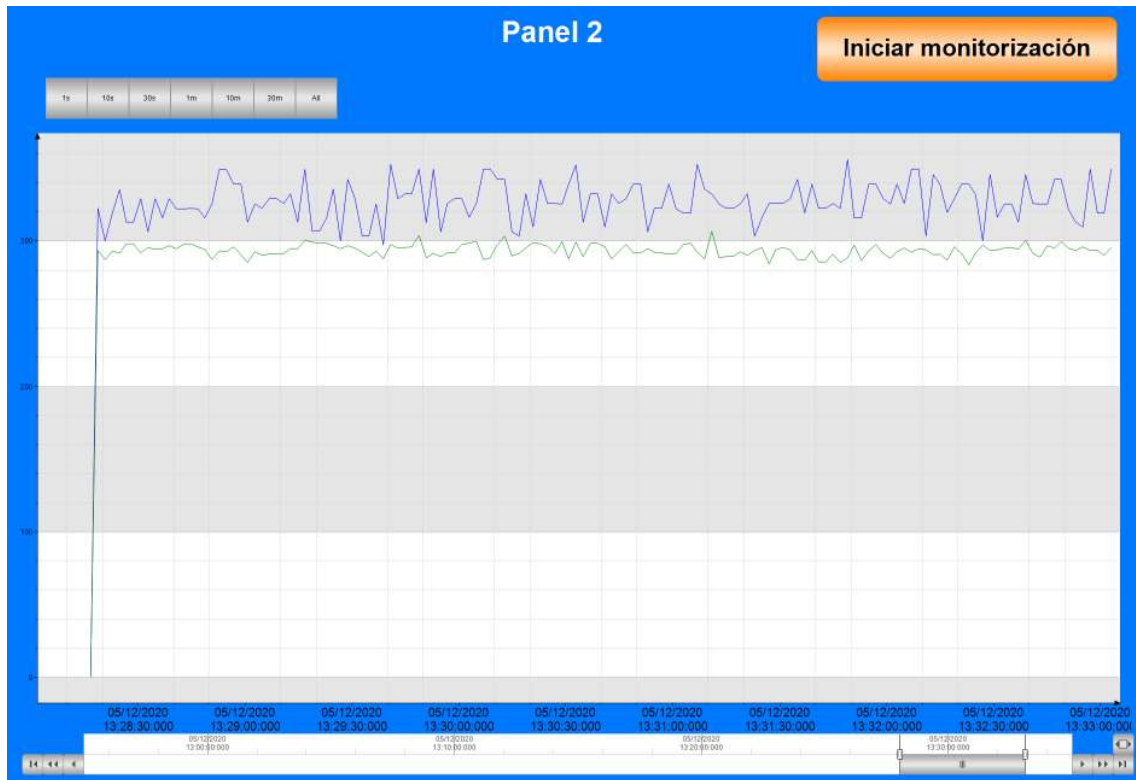


Figura 50: Ventana individual de monitorización

En la parte inferior de cada ventana de monitorización individual se ilustra mediante una barra de estado el valor de la potencia instantánea del panel en cuestión. Los colores de la barra (rojo, amarillo y verde) representan los rangos de valores para fallo, warning y funcionamiento correcto respectivamente acorde al algoritmo de control.

Para volver a la ventana general de monitorización desde cualquier ventana de monitorización individual se dispone de un botón en la parte inferior izquierda.



Figura 51: Ventana individual de monitorización zona inferior

9. Presupuesto

A continuación, el desglose del presupuesto correspondiente a un único sistema de mediciones al que se le asignan 10 paneles fotovoltaicos para el ejemplo de este trabajo.

El incremento de módulos fotovoltaicos a supervisar no incrementa proporcionalmente el presupuesto, ya que la inversión en mano de obra de ingeniería que supone más del 70% del importe total, no incrementaría proporcionalmente. Esto se debe a que, una vez desarrollado el proyecto, la expansión de este lleva pocas horas de labor en comparación con las horas requeridas para desarrollar la idea e implementarla.

PRESUPUESTO						
Medida	Unidades	Descripción	Precio (€)	Unitario	Precio (€)	Total
1	Uds.	Arduino UNO	22,8		22,8	
1	Uds.	Tarjeta de red Ethernet W5100	12,35		12,35	
1	Uds.	Expansión entradas analógicas	18,6		18,6	
10	Uds.	Transductor voltaje CELSA DMU	32,7		327	
10	Uds.	Transductor corriente CELSA DMI	34,3		343	
4	Metros	Cable de red	5,25		21	
7	Metros	Cableado sistema mediciones	4,75		33,25	
1	Uds.	Fuente de alimentación 12 VDC, 50W	18,9		18,9	
120	Uds.	Mano de obra ingeniería	50		6000	
33	Uds.	Mano de obra instalación	30		990	
TOTAL					7786,9	

Figura 52: Desglose presupuesto

El presupuesto total para llevar a cabo el proyecto asciende a 7786,9 euros.

10. Conclusiones

El trabajo realizado propone una posible manera de detectar anomalías en la generación fotovoltaica de cada panel por separado, con el fin de resolver uno de los problemas más frecuentes en las instalaciones fotovoltaicas, el fenómeno punto caliente (hotspot), empleando una tecnología de coste mínimo, pero con las prestaciones suficientes para los requerimientos de la aplicación.

Por otra parte, todos los softwares utilizados son totalmente gratuitos y a disposición de todo usuario, además de que son genéricos y con disponibilidad suficiente de librerías y material aportado por los distintos usuarios que forman las comunidades de usuarios para cada software.

El broker utilizado para implementar el protocolo de comunicaciones MQTT también es totalmente gratuito.

En cuanto al hardware necesario es de lo más económico del mercado, sin embargo, cuenta con buenas prestaciones para esta aplicación. El coste total del hardware no supera los 50 euros por panel fotovoltaico en total (excluyendo los honorarios correspondientes a recursos humanos), por tanto, podemos estar hablando de un proyecto low-cost.

El algoritmo implementado para la detección de anomalías en el funcionamiento de los paneles solares fotovoltaicos es sencillo al mismo tiempo que eficaz.

Uno punto más a destacar de este proyecto es el uso del protocolo de comunicaciones MQTT y el ámbito del IoT que sin duda alguna ha levantado mucha expectativa en los últimos años a nivel mundial.

11. Trabajos futuros

Como posibles trabajos futuros para este proyecto se presentan varias propuestas. En primer lugar, se propone implementar la desconexión automática de los paneles cuyo funcionamiento resulte anormal, es decir, los paneles cuya generación es inferior al valor definido por el algoritmo de control.

Se ha programado una salida digital del sistema de mediciones para ser activada y desactivada desde el sistema de monitorización a modo de comprobación de la eficacia del sistema diseñado a la hora de hacer fluir la información en el sentido inverso (desde el sistema de control hacia el sistema de mediciones), un posible trabajo futuro es dar uso a esta función.

Otro punto a reforzar en el proyecto es la profundización en el algoritmo de diagnóstico preventivo con experimentos físicos y análisis matemático para definir de forma más acertada la desviación exacta que representa el límite del control predictivo. Esto incrementaría la precisión y la eficacia del sistema asegurando no cruzar la línea que separa el control predictivo del control correctivo, ya que el objetivo principal del proyecto es evitar daños en los paneles solar que provoca principalmente el fenómeno hotspot (punto caliente).

Finalmente, se propone enfocar el análisis en los valores de voltaje y de corriente, haciendo un previo estudio profundizando en el comportamiento de estas dos variables por separado ante un sombreado parcial o suciedad en lugar de simplificar el estudio a la variable de potencia como es el caso de este trabajo. Un estudio detallado de las variaciones de voltaje y de corriente por separado nos permitiría desarrollar un algoritmo de control más preciso, y por tanto, mejoraría la eficacia del sistema en su conjunto



12. Referencias

- [1] Red Eléctrica de España (REE): <https://www.ree.es/es>
- [2] Apuntes asignatura Redes y Sistemas distribuidos para control
- [3] Apuntes asignatura Sistemas de control en red. Supervisión con herramientas SCADAS
- [4] Arduino IDE: <https://www.arduino.cc/en/main/OldSoftwareReleases>
- [5] Referencias Arduino: <https://www.arduino.cc/reference/en/>
- [6] Softwares Codesys: https://store.codesys.com/?__store=en
- [7] Referencias Codesy: <https://help.codesys.com/>
- [8] Librería MQTT para Arduino: <https://github.com/256dpi/arduino-mqtt>
- [9] Librería MQTT para Codesys: <https://github.com/stefandreyer/CODESYS-MQTT>
- [10] Aspectos teóricos hotspot: <https://pv-magazine-usa.com>
- [11] Conceptos teóricos y definiciones: <https://es.wikipedia.org/>
- [12] Detection and analysis of hot-spot formation in solar cells. Michael Simon, Edson L. Meyer
- [13] Energética vol.35 no.3 La Habana sep.-dic. 2014
- [14] Análisis de la degradación de módulos fotovoltaicos. Francisco Jesús Reguera Gil
- [15] Artículos científicos relacionados: <https://www.sciencedirect.com/>



Anexo I: Códigos y programas

Código programa Arduino

```
//----- Broker MQTT -----//  
const char* broker = "myarduinobroker.cloud.shiftr.io";  
char mqttUserName[] = "myarduinobroker";  
char mqttPass[] = "9876543210";  
  
//----- Variables y librerías -----//  
#include <Ethernet.h>  
#include <MQTTClient.h>  
EthernetClient net;  
MQTTClient client;  
  
#define MAX_VOLTAGE 150.0 // Rango medición sensor voltaje (0 a 150V).  
#define MAX_CURRENT 15.0 // Rango medición sensor corriente (0 a 15A).  
#define T 2.0 // Tiempo de muestreo en segundos.  
const unsigned long mqttPostingInterval = T * 1000; // Publicar datos cada 'T' segundos.  
unsigned long lastUploadedTime = 0;  
  
//----- ARDUINO 1 -----//  
//----- MAC Arduino1 -----//  
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};  
  
//----- Configuración PIN indicador de conexión establecida -----//  
#define MQTT_CONNECTION_LED_PIN 0 // Indicador conexión MQTT.  
#define OUT1_PIN 7 // Salida de reserva.  
  
//----- Definir TOPICS -----//  
//----- Topics para publicar las potencias de los paneles del Arduino nº1 -----//  
const char* topic_pub_Arduino1_Panel_1 = "Arduino1_Potencia_Panel_1";  
const char* topic_pub_Arduino1_Panel_2 = "Arduino1_Potencia_Panel_2";
```

```

const char* topic_pub_Arduino1_Panel_3 = "Arduino1_Potencia_Panel_3";
const char* topic_pub_Arduino1_Panel_4 = "Arduino1_Potencia_Panel_4";
const char* topic_pub_Arduino1_Panel_5 = "Arduino1_Potencia_Panel_5";
const char* topic_pub_Arduino1_Panel_6 = "Arduino1_Potencia_Panel_6";
const char* topic_pub_Arduino1_Panel_7 = "Arduino1_Potencia_Panel_7";
const char* topic_pub_Arduino1_Panel_8 = "Arduino1_Potencia_Panel_8";
const char* topic_pub_Arduino1_Panel_9 = "Arduino1_Potencia_Panel_9";
const char* topic_pub_Arduino1_Panel_10 = "Arduino1_Potencia_Panel_10";
// Topic para leer publicaciones de Codesys //
const char* topic_sub_CoDeSys = "CoDeSys";

////=====conectar =====////
void connect() {
    digitalWrite(MQTT_CONNECTION_LED_PIN, LOW); // Apagar LED comunicaci3n
    //--- crear cliente ---//
    char clientID[] = "ArduinoBroker_0000000000"; // Generar ID random para cliente.
    for (int i = 9; i < 19 ; i++) clientID[i] = char(48 + random(10));
    while (!client.connect("Arduino1", mqttUserName, mqttPass)) {
        digitalWrite(MQTT_CONNECTION_LED_PIN, !digitalRead(MQTT_CONNECTION_LED_PIN));
        delay(1000);
    }
    digitalWrite(MQTT_CONNECTION_LED_PIN, HIGH); // Encender indicador de comunicaci3n.

    //----- Suscribirse a las publicaciones de Codesys -----//
    client.subscribe(topic_sub_CoDeSys);
}

////===== messageReceived =====////
void messageReceived(String &topic, String &payload) {

    // Activar/desactivar salida de reserva
    if (topic == topic_sub_CoDeSys) {
        int n = payload.toInt();

```



```
    digitalWrite(OUT1_PIN, n);
}
}

////===== setup =====////
void setup() {
    pinMode(MQTT_CONNECTION_LED_PIN, OUTPUT);
    Ethernet.begin(mac);
    client.begin(broker, net);
    client.onMessage(messageReceived);
    connect();
}

////===== loop =====////
void loop() {
    client.loop();
    if (!client.connected())connect();

    //---- Actualizar MQTT ----//
    if (millis() - lastUploadedTime > mqttPostingInterval) {

        //----- Leer sensores panel 1, calcular potencia y enviar resultado al brocker -----//
        float A1_Voltaje_1 = analogRead(A0) / 1023.0 * MAX_VOLTAGE;
        float A1_Corriente_1 = analogRead(A1) / 1023.0 * MAX_CURRENT;
        float A1_Potencia_1 = A1_Voltaje_1 * A1_Corriente_1;

        client.publish(topic_pub_Arduino1_Panel_1, String(A1_Potencia_1), true, 1);

        //----- Leer sensores panel 2, calcular potencia y enviar resultado al brocker -----//  PROBAR CON
        VALORES RANDOM
        float A1_Voltaje_2 = A1_Voltaje_1 * random(95,105)/100.0;
        float A1_Corriente_2 = A1_Corriente_1 * random(95,105)/100.0;
        float A1_Potencia_2 = A1_Voltaje_2 * A1_Corriente_2;
```

```
client.publish(topic_pub_Arduino1_Panel_2, String(A1_Potencia_2), true, 1);
```

```
//----- Leer sensores panel 3, calcular potencia y enviar resultado al brocker -----//  PROBAR CON VALORES RANDOM
```

```
float A1_Voltaje_3 = A1_Voltaje_1 * random(50,70)/100.0;
```

```
float A1_Corriente_3 = A1_Corriente_1 * random(50,70)/100.0;
```

```
float A1_Potencia_3 = A1_Voltaje_3 * A1_Corriente_3;
```

```
client.publish(topic_pub_Arduino1_Panel_3, String(A1_Potencia_3), true, 1);
```

```
//----- Leer sensores panel 4, calcular potencia y enviar resultado al brocker -----//  PROBAR CON VALORES RANDOM
```

```
float A1_Voltaje_4 = A1_Voltaje_1 * random(95,105)/100.0;
```

```
float A1_Corriente_4 = A1_Corriente_1 * random(95,105)/100.0;
```

```
float A1_Potencia_4 = A1_Voltaje_4 * A1_Corriente_4;
```

```
client.publish(topic_pub_Arduino1_Panel_4, String(A1_Potencia_4), true, 1);
```

```
//----- Leer sensores panel 5, calcular potencia y enviar resultado al brocker -----//  PROBAR CON VALORES RANDOM
```

```
float A1_Voltaje_5 = A1_Voltaje_1 * random(95,105)/100.0;
```

```
float A1_Corriente_5 = A1_Corriente_1 * random(95,105)/100.0;
```

```
float A1_Potencia_5 = A1_Voltaje_5 * A1_Corriente_5;
```

```
client.publish(topic_pub_Arduino1_Panel_5, String(A1_Potencia_5), true, 1);
```

```
//----- Leer sensores panel 6, calcular potencia y enviar resultado al brocker -----//  PROBAR CON VALORES RANDOM
```

```
float A1_Voltaje_6 = A1_Voltaje_1 * random(95,105)/100.0;
```

```
float A1_Corriente_6 = A1_Corriente_1 * random(95,105)/100.0;
```

```
float A1_Potencia_6 = A1_Voltaje_6 * A1_Corriente_6;
```

```
client.publish(topic_pub_Arduino1_Panel_6, String(A1_Potencia_6), true, 1);
```

```
//----- Leer sensores panel 7, calcular potencia y enviar resultado al brocker -----//  PROBAR CON VALORES RANDOM
```

```
float A1_Voltaje_7 = A1_Voltaje_1 * random(70,90)/100.0;
```

```
float A1_Corriente_7 = A1_Corriente_1 * random(70,90)/100.0;
```

```
float A1_Potencia_7 = A1_Voltaje_7 * A1_Corriente_7;
```

```
client.publish(topic_pub_Arduino1_Panel_7, String(A1_Potencia_7), true, 1);
```

```
//----- Leer sensores panel 8, calcular potencia y enviar resultado al brocker -----//  PROBAR CON VALORES RANDOM
```

```
float A1_Voltaje_8 = A1_Voltaje_1 * random(95,105)/100.0;
```

```
float A1_Corriente_8 = A1_Corriente_1 * random(95,105)/100.0;
```

```
float A1_Potencia_8 = A1_Voltaje_8 * A1_Corriente_8;
```

```
client.publish(topic_pub_Arduino1_Panel_8, String(A1_Potencia_8), true, 1);
```

```
//----- Leer sensores panel 9, calcular potencia y enviar resultado al brocker -----//  PROBAR CON VALORES RANDOM
```

```
float A1_Voltaje_9 = A1_Voltaje_1 * random(95,105)/100.0;
```

```
float A1_Corriente_9 = A1_Corriente_1 * random(95,105)/100.0;
```

```
float A1_Potencia_9 = A1_Voltaje_9 * A1_Corriente_9;
```

```
client.publish(topic_pub_Arduino1_Panel_9, String(A1_Potencia_9), true, 1);
```

```
//----- Leer sensores panel 10, calcular potencia y enviar resultado al brocker -----//  PROBAR CON VALORES RANDOM
```

```
float A1_Voltaje_10 = A1_Voltaje_1 * random(95,105)/100.0;
```

```
float A1_Corriente_10 = A1_Corriente_1 * random(95,105)/100.0;
```

```
float A1_Potencia_10 = A1_Voltaje_10 * A1_Corriente_10;
```

```
client.publish(topic_pub_Arduino1_Panel_10, String(A1_Potencia_10), true, 1);
```

```
lastUploadTime = millis();
```

```
}
```

```
}
```

Programa Codesys

```
PROGRAM GRUPO_1
```

```
VAR
```

```
    //      Panel fotovoltaico 1
    Arduino1_Panel_1: MQTT_Client.FB_MQTTClient;
    Potencia1: STRING; // Última lectura (STRING)
    Potencia_1_REAL:REAL; // Última lectura W(REAL)

    //      Panel fotovoltaico      2
    Arduino1_Panel_2: MQTT_Client.FB_MQTTClient;
    Publicaciones_2: STRING;
    Potencia2: STRING; // Última lectura (STRING)
    Potencia_2_REAL:REAL; // Última lectura W(REAL)

    //      Panel fotovoltaico      3
    Arduino1_Panel_3: MQTT_Client.FB_MQTTClient;
    Publicaciones_3: STRING;
    Potencia3: STRING; // Última lectura (STRING)
    Potencia_3_REAL:REAL; // Última lectura W(REAL)

    //      Panel fotovoltaico      4
    Arduino1_Panel_4: MQTT_Client.FB_MQTTClient;
    Publicaciones_4: STRING;
    Potencia4: STRING; // Última lectura (STRING)
    Potencia_4_REAL:REAL; // Última lectura W(REAL)

    //      Panel fotovoltaico      5
    Arduino1_Panel_5: MQTT_Client.FB_MQTTClient;
    Publicaciones_5: STRING;
    Potencia5: STRING; // Última lectura (STRING)
    Potencia_5_REAL:REAL; // Última lectura W(REAL)
```




```
//      Panel fotovoltaico      6
Arduino1_Panel_6: MQTT_Client.FB_MQTTClient;
Publicaciones_6: STRING;
Potencia6: STRING; // Última lectura (STRING)
Potencia_6_REAL:REAL; // Última lectura W-REAL)
```

```
//      Panel fotovoltaico      7
Arduino1_Panel_7: MQTT_Client.FB_MQTTClient;
Publicaciones_7: STRING;
Potencia7: STRING; // Última lectura (STRING)
Potencia_7_REAL:REAL; // Última lectura W-REAL)
```

```
//      Panel fotovoltaico      8
Arduino1_Panel_8: MQTT_Client.FB_MQTTClient;
Publicaciones_8: STRING;
Potencia8: STRING; // Última lectura (STRING)
Potencia_8_REAL:REAL; // Última lectura W-REAL)
```

```
//      Panel fotovoltaico      9
Arduino1_Panel_9: MQTT_Client.FB_MQTTClient;
Publicaciones_9: STRING;
Potencia9: STRING; // Última lectura (STRING)
Potencia_9_REAL:REAL; // Última lectura W-REAL)
```

```
//      Panel fotovoltaico 10
Arduino1_Panel_10: MQTT_Client.FB_MQTTClient;
Publicaciones_10: STRING;
Potencia10: STRING; // Última lectura (STRING)
Potencia_10_REAL:REAL; // Última lectura W-REAL)
```

```
Iniciar_monitorizacion: BOOL; // Variable para iniciar/detener monitorización
VAR_OUT_1: BOOL; // Variable de salida reservada para activar/desactivar salida arduino
```

END_VAR

VAR_INPUT

```
// Potencia media grupo paneles 1
Potencia_Media_1:REAL;

// Variable para publicar (STRING)
Publicaciones_1: STRING;
```

END_VAR

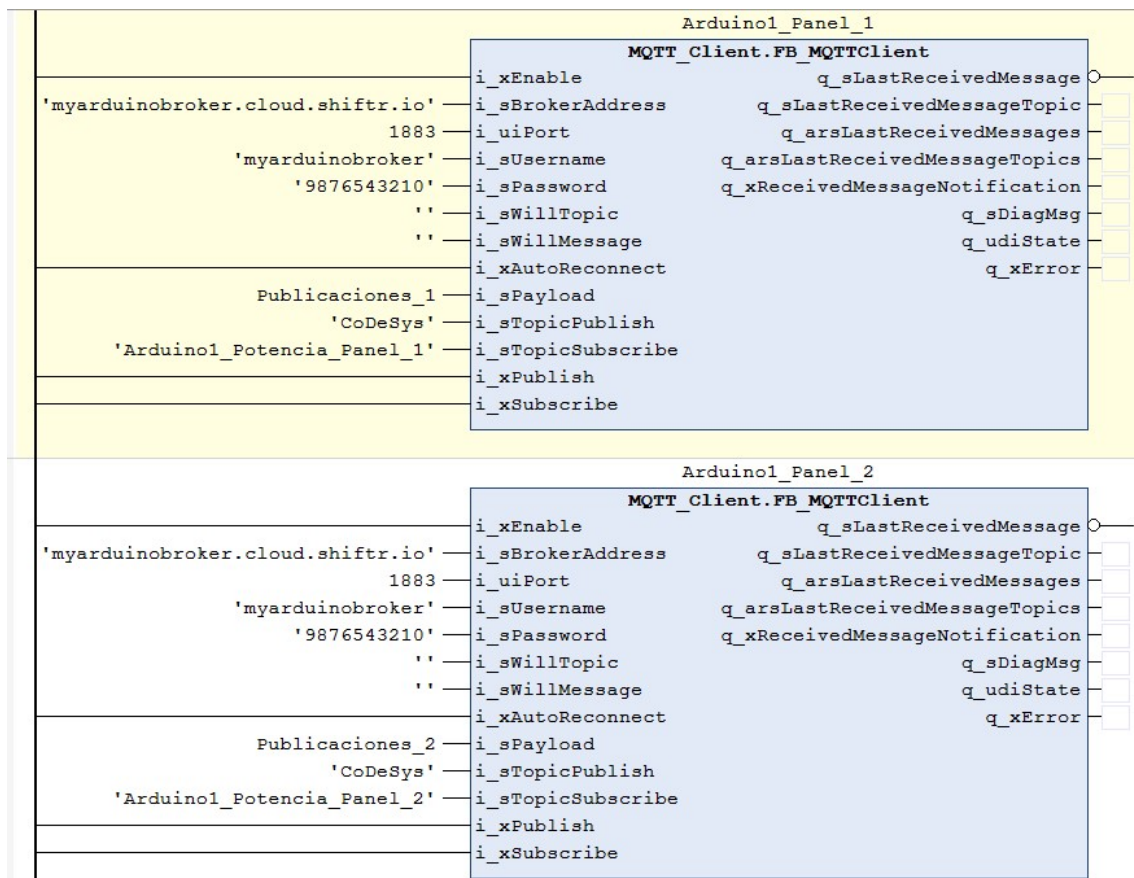


Figura 53: Bloques MQTT_Client Codesys

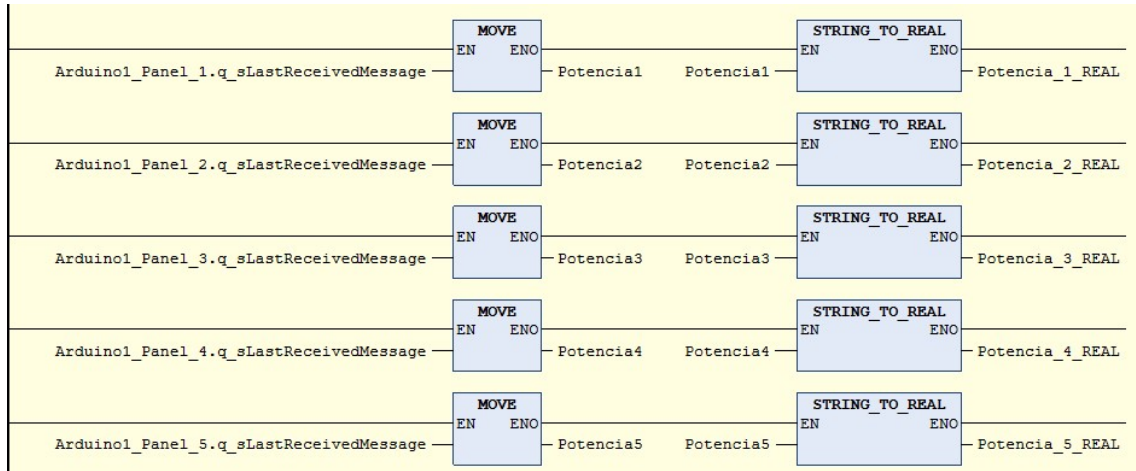


Figura 54: Convertir mensaje de texto leído (REAL) a dato numérico (REAL)

Figura 55: Configurar gráficas

Anexo II: Información técnica del hardware utilizado

Arduino UNO

Arduino Uno es una board basada en un microcontrolador Atmega328. Tiene 14 pines de entrada/salida digital (de los cuales 4 pueden ser utilizados para salidas PWM), 6 entradas análogas, un resonador cerámico de 16 MHz, un conector para USB tipo hembra, un Jack para fuente de Poder, un conector ICSP y un botón reset. Para programar la board se necesita el IDE Arduino.

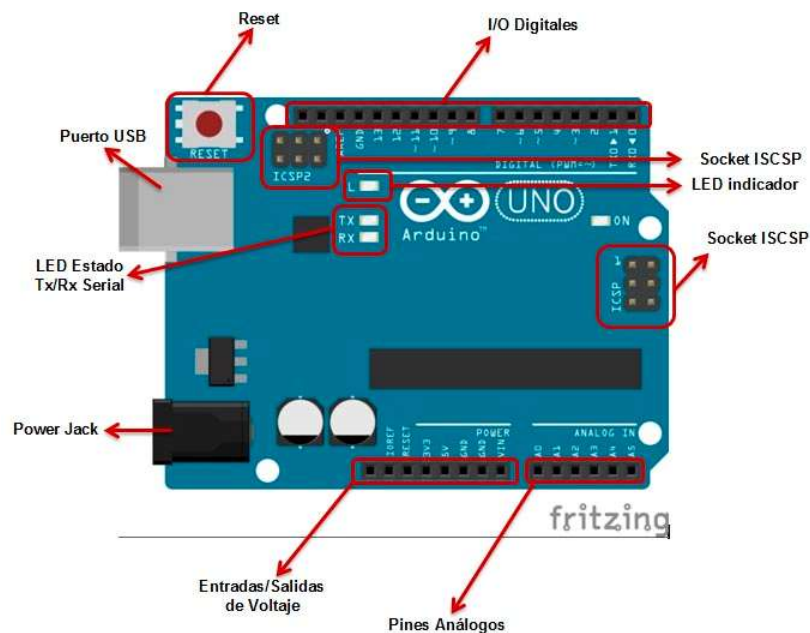


Figura 56: Descripción tarjeta Arduino UNO

- Microcontrolador: ATmega328
- Voltaje Operativo: 5v
- Voltaje de Entrada (Recomendado): 7 – 12 v
- Pines de Entradas/Salidas Digital: 14 (De las cuales 6 son salidas PWM)
- Pines de Entradas Análogas: 6
- Memoria Flash: 32 KB (ATmega328) de los cuales 0,5 KB es usado por Bootloader.
- SRAM: 2 KB (ATmega328)
- EEPROM: 1 KB (ATmega328)
- Velocidad del Reloj: 16 MHZ.

Ethernet Shield W5100

El Shield Ethernet W5100 es una herramienta que abre un sin fin de formas para controlar el Arduino a través de Internet o de la LAN. Domótica, automatización, Internet de las cosas (IoT),

control y monitoreo remoto, etc, son algunos de los campos donde se puede utilizar este shield. Es compatible con Arduino Uno, Mega y Leonardo. Además, las librerías Ethernet y SD vienen incluidas por defecto en el IDE de Arduino, por lo que no hay necesidad de descargar librerías adicionales.



Figura 57: Tarjeta de red Ethernet W5100

- Voltaje de Operación: 5V DC
- Chip Ethernet: Wiznet W5100
- Velocidad Ethernet: 10/100 Mbps
- Conector RJ45
- Interface: SPI
- Compatible con Arduino Uno, Mega, Leonardo
- Lector MicroSD Card
- MAC address: 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
- byte mac [] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED}

Shield expansión entradas analógicas Valleman

Este Shield utiliza sólo 4 puertos I/O (3 x digital, 1 x analógico) pero añade 24 entrada. Por ello, están disponibles 29 entradas analógicas.

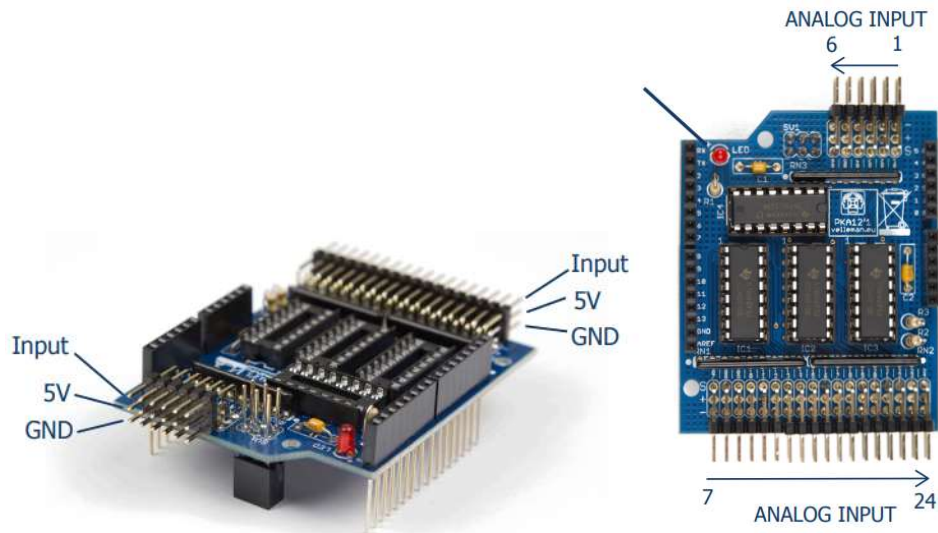


Figura 58: Descripción módulo expansión entradas analógicas

- 24 entradas analógicas
- Utiliza sólo 4 puertos I/O
- Apilable con biblioteca y ejemplos
- Funciona con Arduino UNO™ y placa boards compatibles
- Entradas analógicas: 0 - 5 V DC, 4 - 20 mA
- Dimensiones: 54 x 66 mm (2.1" x 2.6")

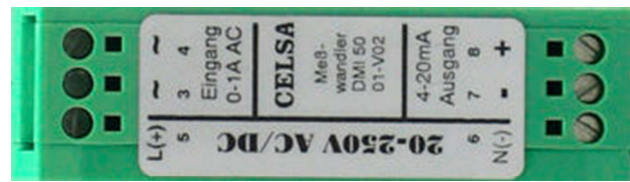
Transductor de voltaje CELSA DMU



Datos Técnicos			
Entrada	Tensión DC (0,01 - 500 V)*	Protección de la caja	IP 30
Salida	Todas las señales estandares**	Consumo max.	2 W
Carga max.	1000 Ω	Anchura	22,5 mm
Alimentación auxiliar	20 - 250 V AC/DC	Longitud	80 mm
Fusible	max. 16 A	Altura	79 mm
Temperatura ambiente	-10 - +65 °C	Material de la caja	PA
Temp. de almacenamiento	-40 - +70 °C	Color	Verde
Cable de conexión	1,5 qmm	Peso	90 - 120 gr.
Protección de los terminales	IP 10	Clase	0,5

Figura 59: Especificaciones transductor de voltaje

Transductor de corriente CELSA DMI



Datos Técnicos			
Entrada	Intensidad DC(0,01 - 15 A)*	Protección de la caja	IP 30
Salida	Todas las señales estandares**	Consumo max.	2 W
Carga max.	750 Ω	Anchura	22,5 mm
Alimentación auxiliar	20 - 250 V AC/DC	Longitud	80 mm
Fusible	max. 16 A	Altura	79 mm
Temperatura ambiente	-10 - +65 °C	Material de la caja	PA
Temp. de almacenamiento	-40 - +70 °C	Color	Verde
Cable de conexión	1,5 qmm	Peso	90 - 120 gr.
Protección de los terminales	IP 10	Clase	0,5

Figura 60: Especificaciones transductor de corriente