Universidad de Alcalá
Escuela Politécnica Superior

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

Trabajo Fin de Grado

Gemelo digital de una planta solar fotovoltaica

Autor: Gema Arribas Fernández

Tutor: Francisco Javier Rodríguez Sánchez

Cotutor: Pablo José Hueros Barrios

2022

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

Grado en electrónica y automática industrial

Trabajo Fin de Grado

Gemelo digital de una planta solar fotovoltaica

Autor: Gema Arribas Fernández

Tutor: Francisco Javier Rodríguez Sánchez

Cotutor: Pablo José Hueros Barrios

TRIBUNAL:

Presidente: DANIEL PIZARRO PÉREZ

Vocal 1º: IGNACIO FERNÁNDEZ LORENZO

Vocal 2º: FRANCISCO JAVIER RODRÍGUEZ SÁNCHEZ

AÑO: 2022

Agradecimientos

Gracias a Francisco Javier y Pablo por enseñarme tanto sobre este tema y por ayudarme con este proyecto.

A mis padres, mi hermano y mi familia, ya que sin ellos no habría conseguido acabar esta etapa académica. Gracias por dedicarme todo vuestro apoyo, esfuerzo y cariño.

A mis amigos, por animarme y sacarme siempre una sonrisa.

1. Índice

1.1. Índice de figuras

1.2. Índice de tablas

2. Resumen en castellano

Los gemelos digitales consisten en modelos ejecutados en tiempo real que permiten un intercambio de datos entre entidades virtuales y sistemas físicos, para mejorar las prestaciones de los sistemas reales, ajustando su control, anticipando posibles problemas y permitiendo ensayar cambios de manera exhaustiva, antes de desplegarlos en el sistema físico. Esta tecnología innovadora es la protagonista de la cuarta revolución industrial y aunque todavía está en su fase de desarrollo, cada vez se usa más en las empresas por sus numerosos beneficios.

Cada día hay más fábricas y domicilios particulares que instalan paneles fotovoltaicos para aprovechar las ventajas de esta energía limpia.

Este proyecto consiste en la unión de ambos conceptos para elaborar la base de un gemelo digital de una planta solar fotovoltaica.

3. Resumen en inglés

Digital twins consist of models executed in real time that allow data to be exchanged between virtual entities and physical systems, in order to improve the performance of real systems, adjusting their control, anticipating possible problems and allowing changes to be tested exhaustively, before deploying them in the physical system. This innovative technology is the protagonist of the fourth industrial revolution and although it is still in its development phase, it is increasingly used in companies due to its numerous benefits.

Every day there are more factories and private homes that install photovoltaic panels to take advantage of this clean energy.

This project consists of the union of both concepts to elaborate the basis of a digital twin of a photovoltaic solar plant.

4. Palabras clave

Gemelo digital
Planta solar fotovoltaica
Matlab
Raspberry Pi
Conexión

5. Resumen extendido

Los gemelos digitales (*Digital Twin*, DT) se basan en un prototipo virtual idéntico a un objeto o proceso físico. El modelo físico debe incluir varios sensores mediante los que se recopilan datos sobre las condiciones de trabajo para transmitirlos al modelo virtual. La entidad virtual procesa los datos obtenidos en tiempo real, con los que, entre otras cosas, puede realizar un mantenimiento predictivo y análisis de fallos. La nueva información obtenida se aplica en el objeto físico para conseguir una mejora en el rendimiento o alargar su ciclo de vida útil. Esta tecnología marca una nueva etapa con el inicio de la industria 4.0. ya que presenta numerosas ventajas a las empresas que optan por este método de análisis de datos en tiempo real. En concreto, en el gemelo digital de una planta solar fotovoltaica, el modelo físico obtiene la energía renovable del sol y el modelo virtual proporciona información para realizar un mantenimiento predictivo, de esta forma si se produce un error se puede actuar con antelación y alarga la vida útil del equipo real.

Para crear el gemelo digital de una planta solar, en este TFG, se construye un modelo virtual mediante dos métodos, cada uno con sus ventajas e inconvenientes. El primer método se basa en Simulink, una Toolbox de Matlab con la que se obtiene el modelo mediante un diagrama de bloques. La ventaja de Simulink es que el modelo creado se puede validar de forma gráfica fácilmente, por ende, la elaboración del mismo es más intuitiva. El segundo método se desarrolla en Matlab con la Toolbox de PV_LIB, la cual proporciona funciones que son las encargadas de modelar el panel fotovoltaico y una base de datos de la que se extraen los parámetros necesarios para la realización del modelo. Para implementar el modelo obtenido se utiliza una Raspberry Pi, puesto que gracias a su bajo coste y reducido consumo de potencia permite que el gemelo digital se esté ejecutando constantemente.

La conexión entre Matlab y la Raspberry Pi se consigue gracias a los paquetes de soporte “Simulink Support Package for Raspberry Pi Hardware” y “MATLAB Support Package for Raspberry Pi Hardware”. Con estos paquetes instalados en Matlab, se puede establecer una conexión de red inalámbrica, consiguiendo que cuando se ejecute un modelo en Simulink sobre la herramienta hardware, se cargue toda la información del modelo automáticamente, sin necesidad de estar pasando los archivos a la Raspberry Pi a través de un pendrive.

Una vez creado el modelo en Matlab, se genera código C que se implementa en la Raspberry Pi, donde se estará ejecutando constantemente el gemelo digital en tiempo real. Se utiliza una placa hardware de bajo coste para implementar el modelo porque prácticamente no consume y así el gemelo digital puede estar funcionando las 24 horas del día, los 365 días del año. Sin embargo, el código que genera Matlab/Simulink hay que modificarlo para cumplir con los objetivos de este proyecto. Las entradas se obtienen de los valores reales medidos en la planta, normalmente disponibles en la nube. Para obtener los datos de la nube, en este TFG, se utiliza la plataforma de ThingSpeak asociada a una planta solar instalada en la Escuela Politécnica de la Universidad de Alcalá de Henares, de la que se obtienen los datos de la irradiancia y la temperatura como parámetros de entrada al gemelo digital. Como salidas tanto del gemelo digital como de la planta fotovoltaica, se consideran la potencia, corriente, tensión, etc. También se estudia el tiempo que tarda en ejecutarse el modelo para analizar su comportamiento en tiempo real.

Las partes más importante de este trabajo, y por lo tanto, a las que se le has dedicado más tiempo, es la conexión entre Matlab y la Raspberry Pi, la búsqueda de un modelo virtual que se asemejase a la planta real y su implementación en código.

6. Memoria

6.1. Introducción

El objetivo principal de este TFG es crear un gemelo digital de una planta solar fotovoltaica. Para poder lograrlo, se tienen en cuenta una serie de propósitos. Por lo tanto, los propósitos de este trabajo son obtener un modelo virtual que represente fielmente el modelo físico, la conversión del modelo a código C, implementar el modelo en un equipo hardware donde se ejecutará y finalmente, analizar las salidas obtenidas de la planta solar y el tiempo de ejecución del modelo para ver si se desarrolla en tiempo real. El modelo se desarrolla gracias a dos Toolbox de Matlab (Simulink y PV_LIB) y la implementación del mismo se lleva a cabo en la Raspberry Pi 4 model B. Para ello, se elabora un gemelo digital de una planta solar fotovoltaica, de forma que se comparen en tiempo real, las salidas del modelo entre los distintos entornos de programación. A partir de la información obtenida del modelo virtual, se puede mejorar el rendimiento de la planta solar, detectando con antelación posibles problemas de funcionamiento de alguno de sus elementos.

Este documento se divide en cuatro capítulos principales. El capítulo 6 es en el que se asientan las bases teóricas y los principios fundamentales para la creación del gemelo digital. En los apartados 6.2 y 6.3 se introduce una descripción teórica de los gemelos digitales de la planta solar fotovoltaica. En el 6.4. se explica el modelo empleado, con una descripción de todos los componentes que lo constituyen. En el apartado 6.5 se presentan algunos de los posibles métodos de comunicación que existen entre el gemelo digital y la planta real, ya que tienen que ser capaces de recibir y enviar información del otro. En los apartados 6.6 y 6.7. se exponen las herramientas software y hardware que se han utilizado en este proyecto. El modelo virtual del gemelo digital se desarrolla en Matlab, pero se han necesitado también otros entornos de programación para llevarlo a cabo. Una vez completado el modelo, se implementa en la Raspberry Pi 4 model B, para que ejecute el gemelo digital en tiempo real. Finalmente, en el apartado 6.8 se han estudiado los métodos de comunicación de la Raspberry Pi con Matlab y Simulink mediante unas Toolboxes de soporte. En el capítulo 7 se describen los métodos empleado para la construcción del gemelo digital, mediante Simulink y PV_LIB. Por último, se muestran los casos de uso y resultados final en el capítulo 8 y las conclusiones y trabajo futuro en el capítulo 9.

6.2. Gemelo digital

Un gemelo digital es una representación virtual de un modelo físico, el cual establece un intercambio de datos bidireccionales en tiempo real. La arquitectura del gemelo digital está formada por tres bloques principales, la sombra digital, el modelo digital y el sistema de control. El primero se trata de la huella digital que deja el gemelo digital y está constituido por una base de datos e interfaces, esta última, se encargan de interactuar con la base de datos e implementar métodos de aprendizaje automático. Esto se necesita para la predicción de datos y para restaurar el modelo de comportamiento del sistema. El segundo se encarga de la operatividad de todo el sistema implementando la interfaz de las interacciones con los demás componentes del modelo digital. El tercero introduce una acción de control en un objeto real, siendo el prototipo el gemelo digital [1].

A continuación, se enumeran las ventajas y desventajas de los gemelos digitales [2]:

Ventajas:

- Detección remota en línea: los gemelos digitales están constituidos por varias tecnologías innovadoras, de forma que se puede recopilar la información del funcionamiento del equipo, a partir de los sensores en tiempo real. Estos datos se analizan con algoritmos de

inteligencia artificial. Una vez analizados, son enviados a una estación remota para que los trabajadores supervisen los resultados y comprueben si se ha producido algún error.

- Predicción de fallas: se comparan los datos actuales con datos históricos para obtener el vector de error residual y umbrales de falla. Con estos parámetros, el DT es capaz de determinar si se producirá algún fallo y en qué momento ocurrirá posiblemente.
- Alta seguridad: para obtener los valores finales y efectuar operaciones en el equipo físico de forma segura y sin correr ningún riesgo, se representan los posibles escenarios en el modelo virtual.
- Amplia aplicación: se puede aplicar la tecnología de los gemelos digitales a cualquier etapa del ciclo de vida del equipo, como la etapa de diseño, etapa de producción y etapa de servicio.
- Reducir los costos de operación y mantenimiento: al realizar las simulaciones en el modelo virtual, se puede predecir el mantenimiento del equipo y el tiempo que conlleva esta tarea. De esta forma, se pueden reducir los constes de operación al mejorar la eficiencia de trabajo.
- Visualización del estado de funcionamiento del equipo: se muestra el proceso de ejecución y los datos del estado del equipo a través de los datos obtenidos de los objetos físicos en tiempo real.

Desventajas:

- El concepto de la tecnología DT no ha sido unificado todavía: la implementación de los modelos de los gemelos digitales es única y diferente entre sí puesto que todavía no hay un modelo unificado. Por este motivo, es muy difícil de integrar un solo modelo y consultar en línea los datos de otros equipos.
- Ejemplos de aplicación limitada: actualmente, la tecnología de los gemelos digitales ha tenido un mayor impulso en las tareas de monitoreo, predicción y diagnóstico de fallas para un solo equipo. Las otras tareas todavía no se han desarrollado significativamente y faltan ejemplos que se puedan consultar de los modelos existentes.
- Se requiere una gran cantidad de sensores: se necesitan numerosos sensores ya que se necesita recopilar una gran cantidad de datos para el seguimiento en tiempo real.
- Gran cantidad de procesamiento de datos: se crean una gran cantidad de datos debido a la comunicación entre los datos del modelo físico y los datos del modelo virtual en tiempo real. Como la cantidad de datos generados es tan grande, el costo de procesamiento es muy elevado y de este problema subyace el inconveniente de simplificar los datos obtenidos, ya que muchas veces los datos están duplicados.

6.2.1. Historia del gemelo digital

El concepto de los gemelos digital se originó en la década de los 60, gracias a la Administración Nacional de Aeronáutica y del Espacio (NASA). Este concepto, nació gracias al programa Apolo de la NASA, en el que se utilizaron dos vehículos espaciales iguales. Uno de ellos, fue el que viajó al espacio y el otro se quedó en la Tierra para mostrar los datos que obtenía de las condiciones de vuelo del vehículo espacial.

Fue en 2002, gracias a Michael Grieves, cuando surgió el término de gemelo digital. Este investigador presentó la idea del gemelo digital como un centro de gestión de ciclos de vida, el cual contaba con una parte física, una parte virtual y la comunicación entre ambas. Sin embargo, y debido a las limitaciones tecnológicas de la época, el gemelo digital se quedó solamente en el concepto. En 2011 apareció el primer artículo en el que se mencionaba al gemelo digital

exponiendo sus ventajas de predicción. En 2012, el concepto de gemelo digital fue establecido por la NASA. Lo definen como una simulación multifísica, multiescala, probabilística y de ultrafidelidad que muestra los datos de un gemelo basado en los datos históricos, los datos del sensor en tiempo real y el modelo físico [3]. En 2014 se publicó el primer White Paper sobre el gemelo digital, y fue desde entonces cuando pasó a estar en su plena fase de crecimiento y desarrollo gracias a la Industria 4.0. Por esto, el gemelo digital evolucionó y pasó de estar compuesto por tres partes a estar compuesto por cinco: una parte física, parte virtual, las conexiones, los datos y los servicios.

6.2.2. Gemelos digitales en sistema de generación solar

Para establecer gemelos digitales de una planta solar fotovoltaica se necesitan un modelo matemático, un método potente de análisis de datos y varias técnicas de inteligencia artificial. El modelo matemático es la parte principal del gemelo digital, porque aprende continuamente de los datos de seguimiento y transforma una cantidad infinita de datos en un grupo fijo de variables del modelo a la vez que valida los datos de monitoreo. Gracias al método de análisis del Big Data se clasifican los datos según las variables del modelo obtenidas con el modelo matemático y suministra información del estado del sistema. Con las técnicas de conocimiento de inteligencia artificial se establece la base de las reglas del conocimiento aplicadas a la información obtenida para generar informes del estado de salud del sistema y poder tomar decisiones en las tareas de mantenimiento [2].

Los gemelos digitales además de monitorear los procesos de fabricación de las células solares para obtener un alto rendimiento y bajo coste, también pueden monitorear y simular unas condiciones específicas de la instalación. De esta forma, se pueden encontrar los problemas de la planta solar real y calcular el tiempo de mantenimiento y reparación del equipo para conseguir aumentar su ciclo de vida. Con la tecnología del internet industrial de las cosas (IIoT) se puede conseguir un mejor rendimiento en la generación de energía solar. Gracias a los sensores del internet de las cosas (IoT) se monitorea la orientación de los paneles solares en tiempo real y con el IIoT se transmiten los datos obtenidos a la nube para su posterior análisis.

6.2.3. Evolución de los gemelos digitales

Según el informe “Digital Twins: Adding Intelligence to the Real World” del Instituto de Investigación Capgemini del año 2022 [4] revela que el 34% de las empresas ya emplean la tecnología de los gemelos digitales a nivel global. Gracias a esto, se consiguen mejorar el rendimiento en un 25% y la sostenibilidad en un 16%, consiguiendo así reducir las emisiones y el consumo de energía. Asimismo, se calcula que, en los próximos cinco años los gemelos digitales aumentarán en un 36%. Como resultado del aumento de esta tecnología revolucionaria, se ha formado un mercado emergente de gemelos digitales. En la figura 1 se puede apreciar como este mercado crece exponencialmente, con más de 50 mil millones de dólares estadounidenses (USD) para el año 2027.

Actualmente, el país más avanzado en este ámbito es Estados Unidos y empresas como el fabricante de coches italiano Maserati, el fabricante de aviones Airbus, la fábrica de Siemens, etc. son las que usan la tecnología de los gemelos digitales.

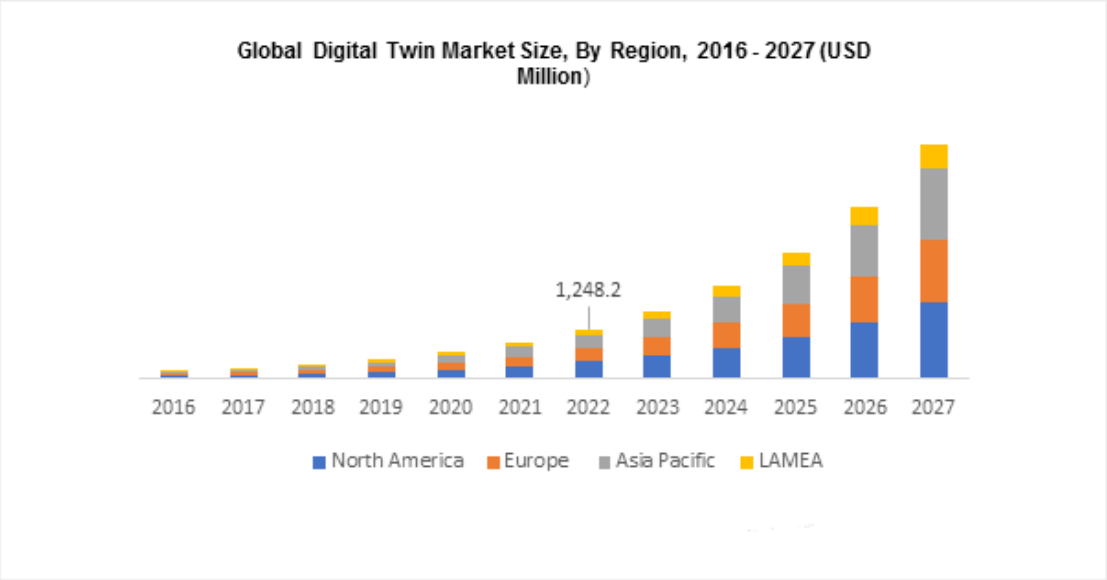

Figura 1: Evolución del mercado mundial de los gemelos digitales

6.2.4. Proyección en el futuro

El actual desarrollo de los gemelos digitales se centra en los siguientes aspectos [2]:

- Desarrollo de sensores integrados avanzados: se necesita un sensor integrado capaz de recolectar toda la información de múltiples sensores en la misma ubicación, puesto que un solo tipo de sensor no es capaz de cubrir las necesidades de monitoreo en línea de la información del gemelo digital en tiempo real.
- Simplificar los datos de cálculo: con el paso del tiempo se generan más datos que analizar, así que es necesario organizar, procesar y almacenar los datos por adelantado, estableciendo bases de datos no relacionales y con métodos de análisis como el Big Data.
- Establecimiento de un marco conceptual del DT unificado: el futuro desarrollo de los gemelos digitales de generación de energía solar consiste en unificar el concepto de DT y el modelo basado en esta tecnología aplicada a varios equipos eléctricos. Se quieren estandarizar varios objetos físicos en los sistemas de generación solar como: la energía del sistema de generación, transmisión y equipo de distribución y sistema de almacenamiento de energía. Asimismo, conseguir la interacción de los datos en el marco del concepto de gemelo digital unificado.
- Trazabilidad direccional del ciclo de vida completo de objetos: el gemelo digital logra la trazabilidad del ciclo de vida completo del objeto físico y predice y analiza el futuro estado de funcionamiento de los equipos reales gracias a la interacción de información en tiempo real entre el modelo físico y el virtual.
- Regulación de retroalimentación para todo el ciclo de vida de objetos físicos: se optimiza el proceso de producción gracias a la retroalimentación en la etapa de diseño mediante la comparación de los valores medidos con los valores históricos. Y en la etapa de operación, se retroalimenta a la etapa de diseño para mejorar el diseño de los futuros productos.

6.3. Planta solar fotovoltaica

6.3.1. Celda solar

La celda solar es el núcleo de los paneles solares fotovoltaicos. Está formada por un material semiconductor mediante el cual, se convierte la radiación solar en energía eléctrica gracias al efecto fotovoltaico como se observa en la figura 2. El material semiconductor que forma la celda es fundamentalmente el silicio, y este puede ser monocristalino o policristalino dependiendo del nivel de eficiencia que se quiera conseguir. El silicio monocristalino se considera de los mejores materiales para construir celdas solares debido a su alta eficacia.

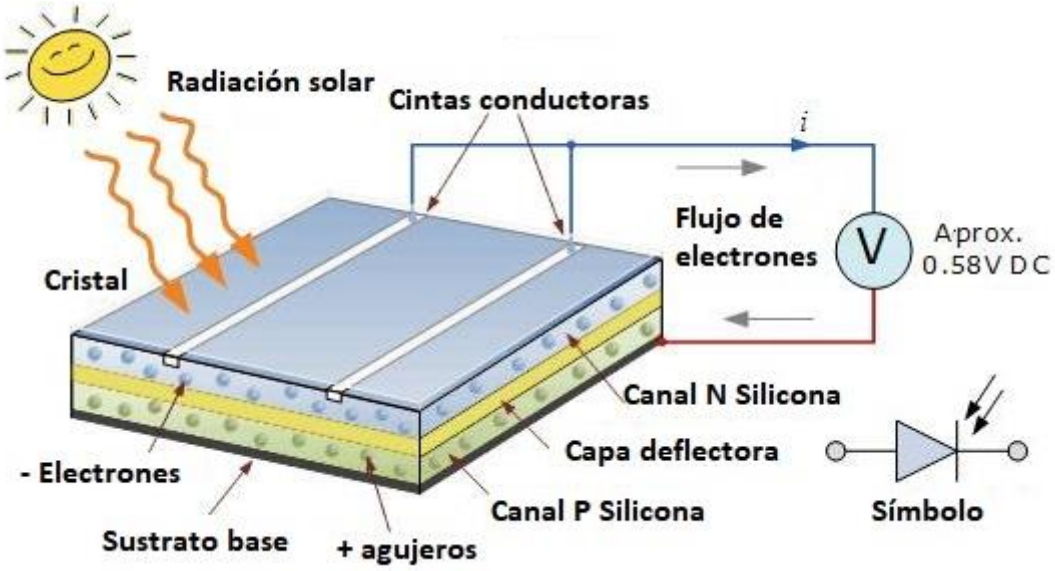

Figura 2: Composición de una célula solar

El efecto que tiene lugar en el interior de la celda solar es el efecto fotovoltaico. Consiste en la generación de corriente eléctrica debido a la exposición de un material a la luz. Se basan en dos zonas, la zona superior de la celda formado por silicio dopado del tipo N (con electrones), y la zona inferior formado por silicio dopado del tipo P (con huecos), ambas zonas inicialmente tienen una carga neutra. Cuando un fotón incide sobre la superficie del material expulsa el electrón que se encontraba en la zona N generando un hueco. Este electrón se desplaza hasta la zona P donde llenará otro hueco disponible (figura 3). Como la zona N pierde electrones se carga positivamente y como la zona P pierde huecos se carga negativamente. Esto provoca que se genere una diferencia de potencial en la zona de unión PN con dirección de N a P. De esta forma, cuando ambas zonas han perdido cierta cantidad de portadores, se genera una barrera que impiden que los electrones se desplacen hasta la banda negativa (zona P) y los huecos hasta la banda positiva (zona N) [5].

Hay que tener en cuenta que el grosor de la capa N es menor que el grosor de la capa P. El grosor de la capa N es pequeño, ya que solamente se necesita para crear la diferencia de potencial generada en la unión PN. Mientras que el grosor de la capa P tiene que ser mayor para que pueda captar todos los fotones posibles sin generar demasiadas recombinaciones de electrón – hueco.

Para que el efecto fotovoltaico se produzca la energía del fotón debe ser mayor que la energía del electrón para que este sea expulsado del material, si la energía no es suficiente el electrón no será capaz de escapar de la superficie, o lo que es lo mismo:

$$\text{Efoton} = h * v \geq Eg$$

Siendo h la constante de Planck ($6{,}626*10^{-34}$ Js) y v la frecuencia del fotón.

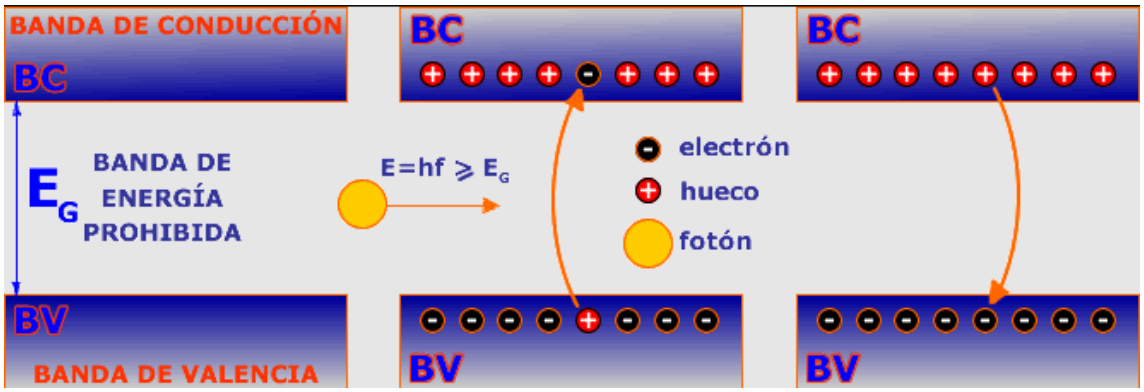

Figura 3: Efecto fotovoltaico

6.3.2. Curvas características de una celda solar

Hay dos parámetros esenciales que determinan las características eléctricas de una celda solar fotovoltaica, la tensión de circuito abierto y la corriente de cortocircuito.

- Tensión de circuito abierto (Voc): es la diferencia de potencial eléctrico que se genera en bornes de un circuito cuando se le desconecta la carga, es decir la corriente es cero.
- Corriente de cortocircuito (Isc): es el valor máximo que alcanza la corriente en un instante de tiempo cuando la celda está en cortocircuito, es decir, la tensión es cero.

Gracias a estos parámetros se pueden definir las curvas de corriente y potencia en función de la tensión. En la figura 4 se pueden apreciar otros parámetros característicos como el punto de máxima potencia (MPP). Este es el punto de trabajo en el que el panel solar genera la máxima potencia (P_M), por lo que se definen la corriente de máxima potencia (I_M) y la tensión de máxima potencia (V_M).

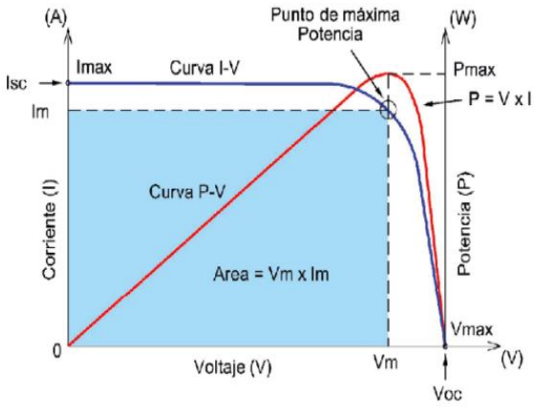

Figura 4: Curva de corriente/potencia de una celda solar en función de la tensión

Estas curvas pueden verse modificadas en función de la temperatura y la irradiancia. Por un lado, cuanto mayor es la irradiancia mayor es la corriente y mayor es la tensión, siendo los incrementos de la tensión menores que los de la corriente. Por otro lado, el aumento de temperatura genera solamente una disminución en la tensión, ya que prácticamente no genera variaciones en la corriente. Ambos efectos se muestran en las figuras 5 y 6 respectivamente.

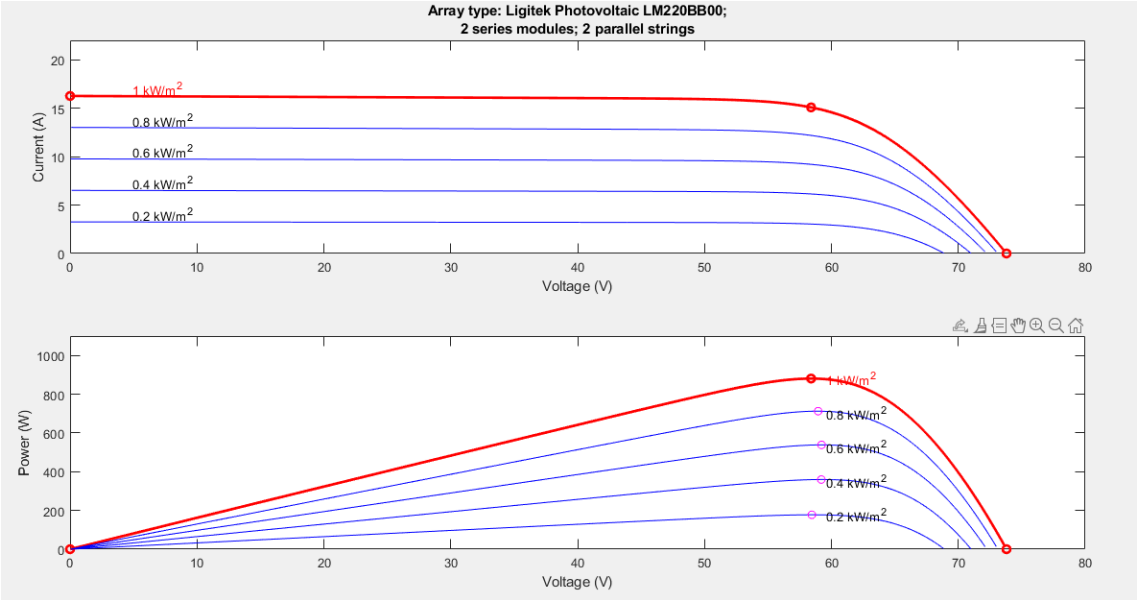

Figura 5: Curvas de corriente/potencia frente a la variación de irradiancia a temperatura constante de 25ºC

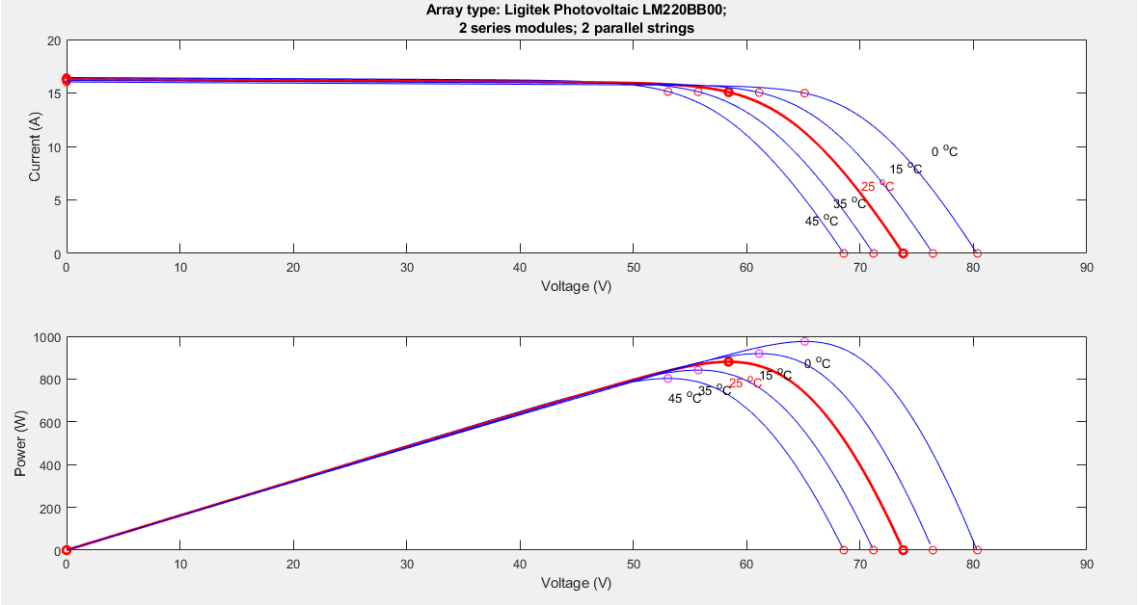

Figura 6: Curvas de corriente/potencia frente a variaciones de temperatura a irradiancia constante de 1000 W/m3

Ante la vista de ambas gráficas, se concluye que cuanto mayor es la irradiancia mayor es la potencia en el punto de máxima potencia. Sin embargo, cuanto mayor es la temperatura, la potencia en el punto MPP es menor.

6.3.3. Modelado de una celda solar

El modelado de una celda solar se basa en un circuito simplificado mediante un generador de corriente y un diodo. A este modelo simplificado hay que añadirle los efectos de pérdida de la corriente de fuga mediante una resistencia en paralelo (RSH) y la pérdida por caída de tensión a la salida del circuito mediante una resistencia en serie (RS). De esta forma se obtiene el circuito eléctrico equivalente de una celda solar realista mostrado en la figura 7.

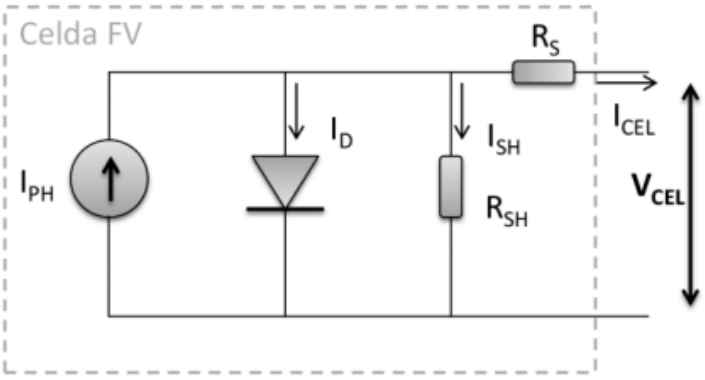

Figura 7: Circuito equivalente de un panel solar

6.3.4. Ecuaciones características de la celda solar.

Siguiendo el esquema eléctrico obtenido del modelado de la celda solar, se concluyen las siguientes ecuaciones [6]:

Corriente fotovoltaica (Icel): corriente generada por la celda solar

$$Icel = I_{PH} - I_O \left[\exp\left(\frac{q(V+Icel*R_S)}{K*Tc}\right) - 1\right] - \frac{V+Icel*R_S}{R_{sh}} \quad (1)$$

Fotocorriente (I_{PH}): o fotocorriente generada, la cual varía dependiendo de las condiciones de trabajo de la celda, es decir, la irradiancia y la temperatura.

$$I_{PH} = \left[I_{SC} + U_{Isc}\left(Tc - Tc_{ref}\right)\right] * \frac{G}{1000} \quad (2)$$

Corriente de saturación inversa (Io)

$$I_O = I_{rs}\left(\frac{Tc}{Tc_{ref}}\right)^3 \exp\left[\frac{qE_{go}\left(\frac{1}{T_{ref}} - \frac{1}{Tc}\right)}{K*A}\right] \quad (3)$$

Corriente de saturación inversa en condiciones de referencia (Irs)

$$I_{rs} = \frac{I_{SC}}{\exp\left(\frac{Voc}{NsKTcA}\right) - 1} \quad (4)$$

Potencia máxima (P_M): obtenida gracias al producto de la corriente y la tensión en el punto de máxima potencia.

$$P_M = I_M * V_M \quad (5)$$

Factor de forma (FF): relación entre la potencia máxima de la célula solar y la potencia de cortocircuito, sirve para determinar el rendimiento.

$$FF = \frac{P_M}{I_{SC}*V_{OC}} \quad (6)$$

Eficiencia de la conversión o rendimiento: relación entre la máxima potencia y la irradiancia incidente en la célula.

$$\eta = \frac{P_M}{S*G} \quad (7)$$

Corriente del panel (Ipanel): corriente final obtenida a la salida del panel solar fotovoltaico

$$Ipanel = Np * I_{PH} - Np * I_O \left[\exp\left(\frac{q(V+I*R_S)}{K*Tc*A}\right) - 1\right] - \frac{V(\frac{Np}{Ns})+Icel*R_S}{R_{sh}} \quad (8)$$

La ecuación (8) se puede abreviar, debido a que la resistencia en paralelo tiende a infinito y se considera como un circuito abierto, de forma que esta rama no afectaría significativamente a la eficiencia de la celda solar.

$$Ipanel = Np * I_{PH} - Np * I_O \left[\exp\left(\frac{q(V+I*R_S)}{K*Tc*A}\right) - 1\right] \quad (9)$$

Parámetros

- Q: carga del electrón = 1,6021*10^{-19} C
- K: constante de Boltzman = 1,380*10^{-23} J/K
- Tc: temperatura en ºK de la célula
- Tcref: temperatura de referencia = 298ºK
- A: factor de idealidad, normalmente se considera la unidad
- UIsc: coeficiente de temperatura de corriente de cortocircuito = 0,060%/ºC
- G: irradiancia solar W/m^3
- S: área de la célula
- I_M: corriente en el punto de máxima potencia
- V_M: tensión en el punto de máxima potencia
- Ns: número de células en serie de un panel
- Np: número de ramas en paralelo de un panel

6.3.5. Tecnologías de la celda solar

Las celdas solares pueden estar hechas por varios materiales, y se elegirá uno u otro en función de las características de la planta solar [5]. En la figura 8 se pueden apreciar las diferencias visuales entre las tecnologías comentadas a continuación.

- Monocristalina

Tienen un gran rendimiento (del 14% al 16%) y una buena relación potencia – superficie (150Wp/m^2), por lo que se puede ahorrar espacio. Visualmente se las diferencia por estar formadas por celdas fotovoltaicas cuadradas de color azul con los bordes circulares. Su coste es muy elevado ya que están formadas con silicio de gran pureza.

- Policristalina

Cuentan con una eficiencia y una relación potencia – superficie alta pero menor que las celdas monocristalinas (14% y 100Wp/m^2 respectivamente). Visualmente se pueden diferenciar de las otras porque la forma de las células solares es cuadrada. Están compuestas de silicio de menor pureza que las celdas monocristalinas, por lo que tienen un coste menor. Debido a que la eficiencia es similar a la tecnología monocristalina y su bajo coste, son las más comunes en las instalaciones de viviendas particulares.

- Película fina amorfa

Se necesita más espacio que con las celdas monocristalinas y policristalinas para generar la misma cantidad de energía, pero su precio es menor. Una ventaja de esta tecnología es que funciona mejor en condiciones de poca luz, en días nublados y sombreado parcial. Visualmente se las puede diferencias por estar formadas por cintas delgadas de color gris oscuro y por su maleabilidad. Su eficiencia es menor (del 5% al 7%) en comparación con las otras dos tecnologías.

Figura 8: Tecnología de las celdas solares

6.3.6. Panel fotovoltaico

El panel fotovoltaico es el encargado de convertir la irradiancia en energía eléctrica gracias a las células solares y convertirla en corriente continua. Este panel está compuesto por varias asociaciones de células solares en serie y/o paralelo para conseguir las condiciones de tensión y corriente deseadas (normalmente las tensiones estándar de trabajo son de 12V, 24V o 48V). Se establece el número de asociaciones en serie y paralelo que forman el panel en función de la corriente y la tensión deseadas. Primero se determina el voltaje requerido, para ello, se agrupas las celdas solares en serie. Una vez que el voltaje ya se ha determinado, las ramas con las celdas solares en serie se agrupan en paralelo para lograr la corriente necesaria (figura 9). Por lo tanto, para conseguir una potencia específica, dependerá del número de células solares y del tipo de conexión entre ellas.

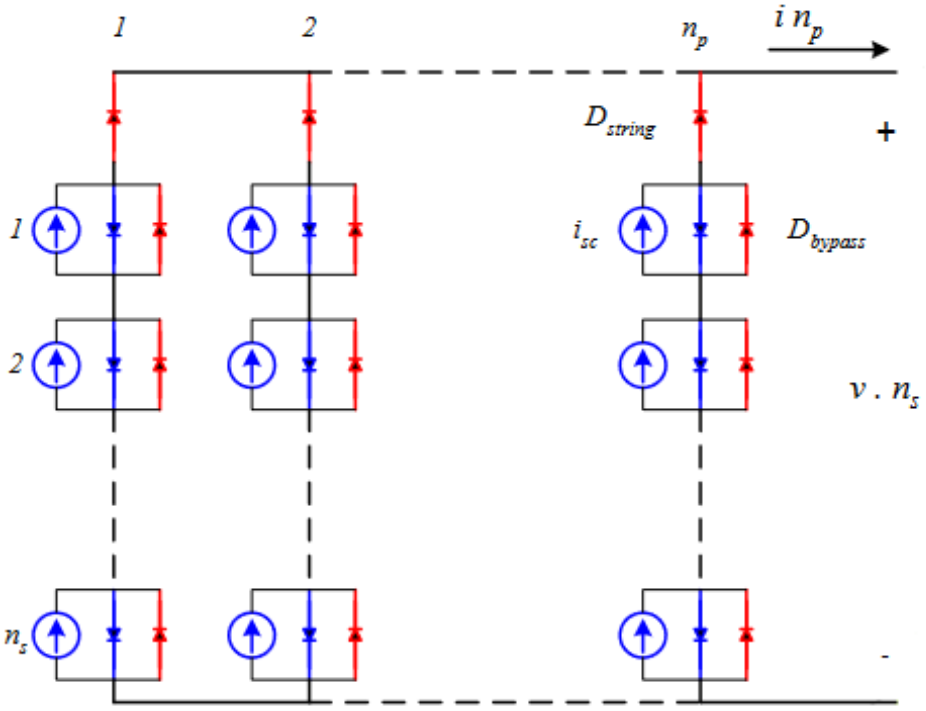

Figura 9: Modelo de un array solar

6.3.7. Elementos de una panel fotovoltaico

En este apartado se describen los diferentes elementos del panel fotovoltaico [7] y en la figura 10 se pueden ver la disposición de los mismos.

- Cubierta de vidrio: está orientada al sol y es la encargada de transmitir la radiación solar a las células fotovoltaicas.
- Encapsulante: material compuesto de EVA (etil – vinil – acetileno) encargado de proteger a las celdas solares y las conexiones entre ellas. Este material es transparente para que la radiación solar captada en la cubierta de vidrio llegue hasta las celdas solares y es un buen aislante de los agentes atmosféricos.
- Cubierta posterior: protege principalmente contra la humedad y contribuye con la estabilidad del sistema.
- Marco de apoyo: es un marco de aluminio que aporta robustez y estanqueidad al conjunto uniendo el módulo con la estructura exterior del panel.
- Cableado y bornas de conexión: se encuentran en la caja estanca de la que salen dos cables (uno positivo y otro negativo) para poder continuar con el circuito eléctrico interno del panel con el exterior.
- Diodo de protección: se encarga de proteger contra sobrecargas.
- Células solares: ya se han estudiado previamente.

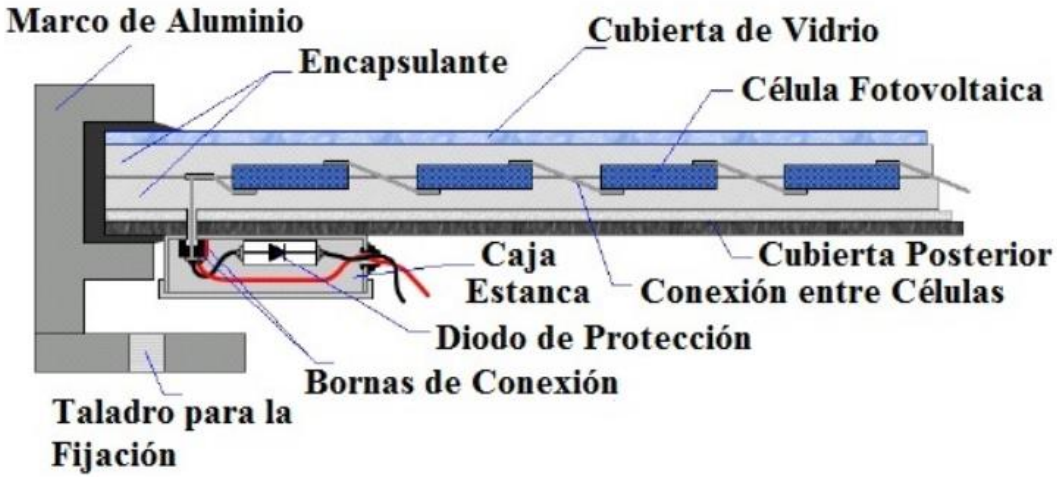

Figura 10: Elementos de un panel fotovoltaico

Todas las características eléctricas, mecánicas, de operación, dimensiones y embalaje vienen especificadas en la hoja técnica del fabricante.

6.3.8. Seguimiento del punto de máxima potencia (MPPT)

El MPPT se encarga de rastrear al punto de máxima potencia (mencionado anteriormente) y de establecer las cargas óptimas según la irradiancia, la sombra y la temperatura. De forma que, ajustando la carga se pueda mantener el panel solar con la potencia al máximo. El MPP es único para unos valores determinados de irradiancia y temperatura. Para alcanzar el punto de máxima potencia existen múltiples algoritmos de seguimiento, algunos de los cuales son:

a. Perturbación y observación (P&O)

Este algoritmo consiste en observar la variación del voltaje de operación en función de la variación de la potencia. Cuando el voltaje de operación se perturba y la potencia aumenta significa que el punto de operación se ha movido hacia el MPP, por lo tanto, el voltaje de operación debe desplazarse en la misma dirección. Sin embargo, si la potencia disminuye significa que el punto de operación se ha movido en sentido contrario al MPP, por lo que, el voltaje de operación debe desplazarse en la dirección opuesta [8]. Se alcanza el punto de máxima potencia cuando el incremento de potencia y de tensión es cero. Mediante ecuaciones, expresar el funcionamiento del algoritmo es el siguiente:

$\Delta V \cdot \Delta P > 0 \rightarrow V_{t+1} = V_t + V_{step}$
$\Delta V \cdot \Delta P < 0 \rightarrow V_{t+1} = V_t - V_{step}$

Este método es el más fácil de implementar, pero puede provocar oscilaciones alrededor del punto de máxima potencia debido a las condiciones atmosféricas.

b. Conductancia incremental (IC)

Este algoritmo se basa en comparar los cambios de la corriente con respecto a la tensión del convertidor. Para hallar el MPP se quiere conseguir que la derivada de la potencia sea cero, es decir, la pendiente de la curva característica I – V sea nula. Cuando el punto se encuentre alejado del punto de máxima potencia mayor será el valor de la división y para alcanzarlo, el algoritmo

calcula hacia donde hay que modificar el punto de trabajo mediante el ajuste de tensión [8]. Este método se expresa mediante la ecuación:

$$\frac{dP}{dV} = I + V * \frac{dI}{dV}$$

Este método es mucho más robusto que el anterior ya que no tienes las oscilaciones cerca del MPP, pero su implementación es más compleja.

c. Voltaje constante (CV)

Este método consiste en mantener la tensión constante ya que la variación de irradiancia a la entrada del módulo solar no le afecta notablemente. Se intenta conseguir que la corriente del módulo solar sea cero, en un breve instante de tiempo, para poder medir la relación entre la tensión de máxima potencia y la tensión de circuito abierto. El funcionamiento de este algoritmo se basa en la expresión:

$$\frac{Vmp}{Voc} = 0{,}76$$

Esta ecuación significa que el punto de máxima potencia se encuentra al 76% de la tensión de circuito abierto. Uno de los inconvenientes de este algoritmo es el desperdicio de energía cuando se desconecta la carga del módulo fotovoltaico, asimismo, el MPP no siempre se encuentra al 76% del voltaje de circuito abierto [9].

6.3.9. Sistemas fotovoltaicos

Existen dos tipos de sistemas fotovoltaicos, el sistema fotovoltaico autónomo y el conectado a la red. La diferencia principal entre ambos sistemas es que el autónomo (figura 11) debe contar obligatoriamente con un regulador para gestionar el almacenamiento de la energía sobrante durante las horas de luz en baterías y así para poder usarla posteriormente cuando la producción no sea suficiente. El sistema conectado a la red (figura 12) cuenta con el soporte de la red eléctrica tanto para suministrar energía durante los déficits de generación como para inyectarla en el caso de superávit. El intercambio energético se registra con un contador que cuantifica la energía demandada a la red eléctrica cuando la producción del sistema es insuficiente y la entregada a la misma cuando se genera un exceso de energía.

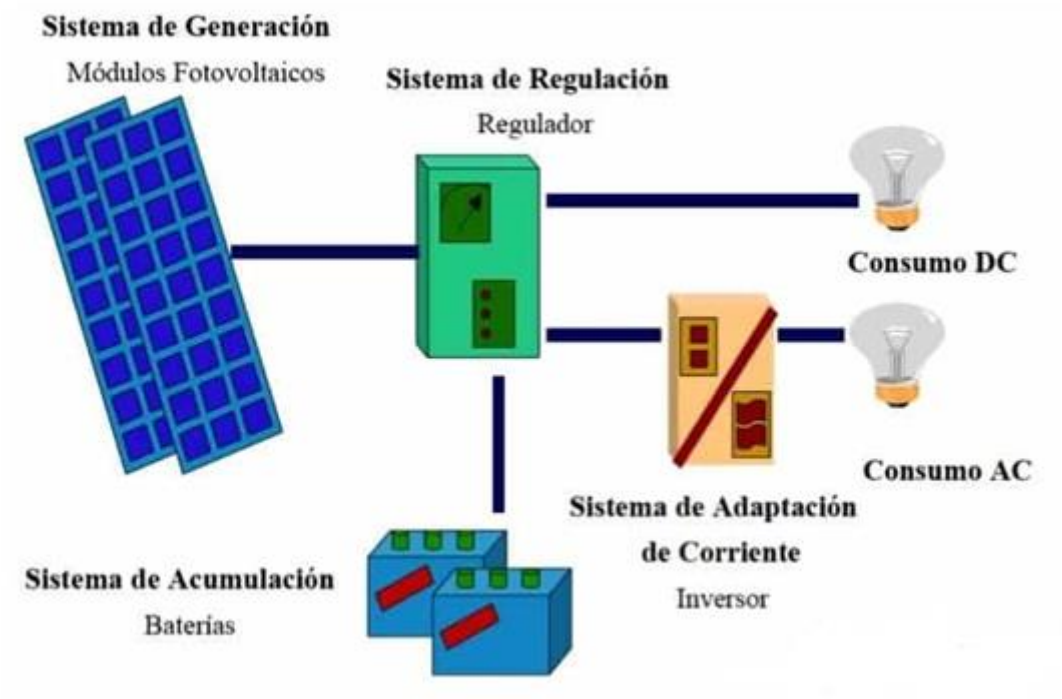

Figura 11: Instalación fotovoltaica autónoma

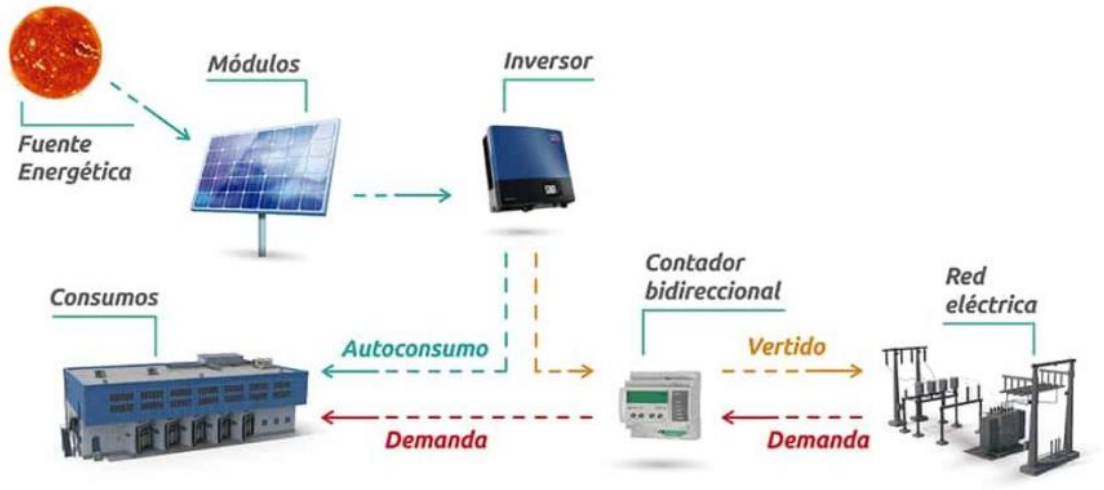

Figura 12: Instalación fotovoltaica conectada a la red eléctrica

Los componentes del sistema fotovoltaico son los siguientes [9]:

a. Baterías

Se disponen en serie o paralelo para almacenar la energía generada y usarla cuando la producción de la misma no sea suficiente. Las funciones principales de las mismas son:

- Proporcionar en un breve periodo de tiempo, corrientes superiores a las generadas en los módulos solares.
- Mantener una tensión de salida constante, puesto que la tensión se ve alterada por la irradiancia y la temperatura.

La vida útil de las baterías se mide en ciclos de carga – descarga. Es importante que la profundidad de descarga (valor de la energía, en porcentaje, de un acumulador cargado en una descarga

completa) no sea demasiado alta porque esto acorta la vida útil de las baterías. En estas baterías se suelen usar tiempos de descarga de 100h [9].

Los tipos de baterías fotovoltaicas que se pueden encontrar son las de níquel – cadmio y las de plomo – ácido. Las baterías de níquel – cadmio son más caras, pero soportan una mayor profundidad de descarga (hasta 90%) y temperaturas muy bajas. Sin embargo, las baterías de plomo – ácido son más económicas y por eso son las más empleadas en sistemas fotovoltaicos. Se suministran en celdas de 2V y soportan altas profundidades de carga (hasta 80%).

b. Regulador

Es el elemento de conexión entre las celdas solares y las baterías. El regulador mide el voltaje de las baterías hasta que llegan a su carga máxima. A continuación, desconecta las baterías del panel solar para que no se sigan cargando y evitar así que se dañen. Posteriormente, los receptores consumen la corriente de las baterías para que el voltaje vuelva a estar en los valores especificados. Finalmente, se vuelven a conectar con el panel solar para conseguir que la corriente de flotación mantenga a las baterías cargadas [9].

c. Inversor

El inversor se encarga de convertir la corriente continua en corriente alterna. Como el panel fotovoltaico genera corriente continua, es necesario para poder conectar los dispositivos electrónicos que trabajan en corriente alterna y en las instalaciones conectadas a la red eléctrica [9]. Los inversores que se conectan a la red eléctrica tienen que trabajar en unas condiciones determinadas (230V y 50Hz) para no generar perturbaciones en la misma.

d. Contador

Es el dispositivo electrónico que mide la energía demandada de la red eléctrica cuando la producida por los paneles fotovoltaicos no es suficiente y la energía cedida cuando hay un exceso de la misma. Se puede usar un contador bidireccional o dos contenedores unidireccionales.

e. Panel solar fotovoltaico

Ya se han estudiado en los apartados anteriores.

6.3.10. Evolución de la energía solar fotovoltaica

Aunque todavía los combustibles fósiles y la energía nuclear son las principales fuentes de energía y suponen el 70% de la generación eléctrica en todo el mundo, el 39% de la potencia instalada en el año 2020 fue de la energía solar fotovoltaica.

La potencia mundial total instalada de energía solar a principios de 2021 es de casi 714GW, siendo China el país con más potencia instalada, puesto que cuenta con 254 GW, o lo que es lo mismo, más del 35% de la capacidad mundial instalada. En Europa, Alemania seguirá siendo el país con la mayor potencia fotovoltaica instalada, España instalará 20 GW de capacidad entre 2019 y 2024, convirtiéndose en el segundo mercado fotovoltaico y le seguirán países como Francia, Países Bajos e Italia [10], como se muestra en la figura 13.

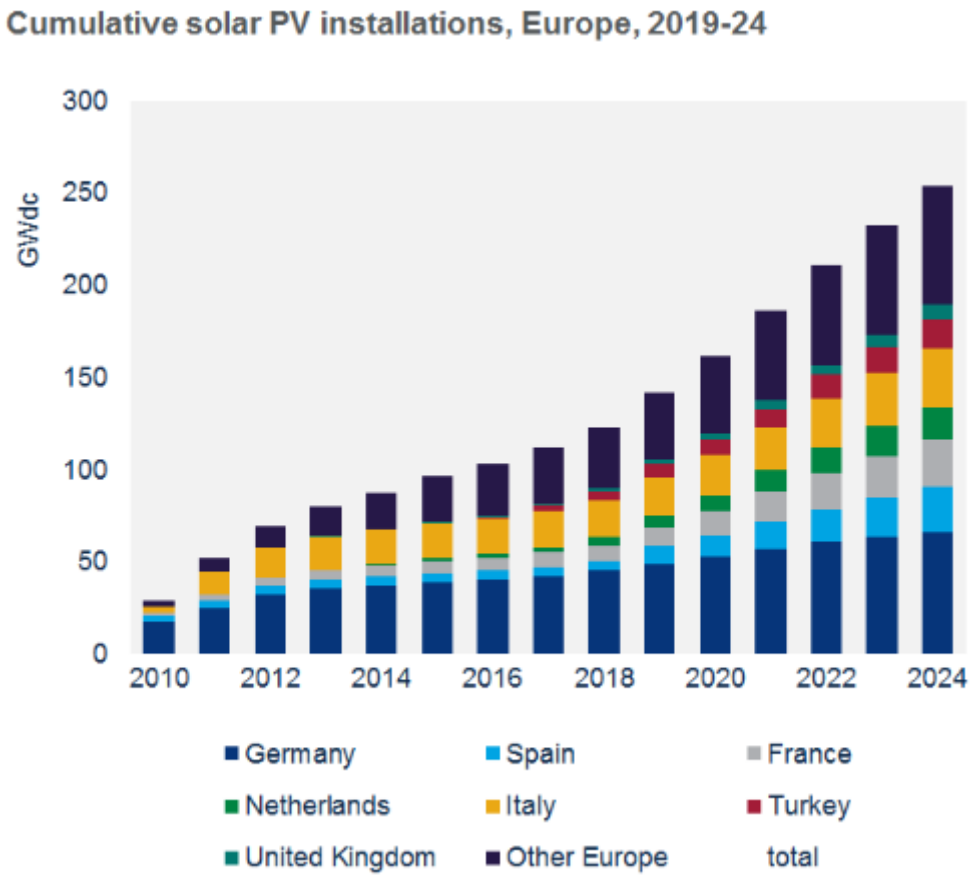

Figura 13: Evolución de los paneles fotovoltaicos instalados en Europa

6.4. Modelo DT de una planta solar fotovoltaica

Para crear un modelo semejante al gemelo digital físico que se encuentra en la Escuela Politécnica de la Universidad de Alcalá, se crea un modelo basado en una instalación fotovoltaica autónoma (figura 14). La cual, contiene el panel solar fotovoltaico, un convertidor reductor DC-DC, una batería para almacenar la energía, un regulador y el algoritmo P&O de seguimiento del punto de máxima potencia

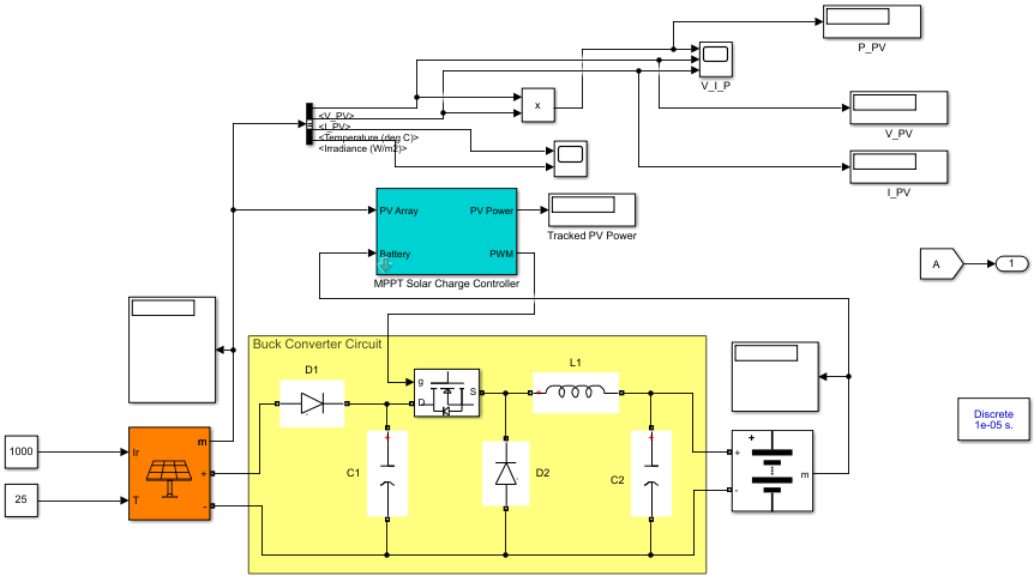

Figura 14: Modelo de la planta solar fotovoltaica completa con el bloque PV array

Funcionamiento y descripción de los parámetros del bloque PV array:

El panel fotovoltaico se encarga de convertir la irradiancia en energía eléctrica. En este bloque se implementa una matriz formada por cadenas de módulos fotovoltaicos conectados en paralelo y cada una consta de módulos conectados en serie. Con el bloque PV array se pueden modelar los módulos fotovoltaicos preestablecidos por el Laboratorio Nacional de Energía Renovable (NREL) o establecer módulos fotovoltaicos definidos por el usuario, en este caso se elige el módulo del NREL "Ligitek Photovoltaic LM220BB00". El bloque cuenta con dos puertos de entrada, uno para la irradiancia en el rango de 0 a 1000 W/m^2 y otro para la temperatura medida en grados Celsius, el cual se habilita al dejar vacía la casilla de "Robust discrete model".

Se pueden observar los parámetros establecidos en este bloque en la figura 15:

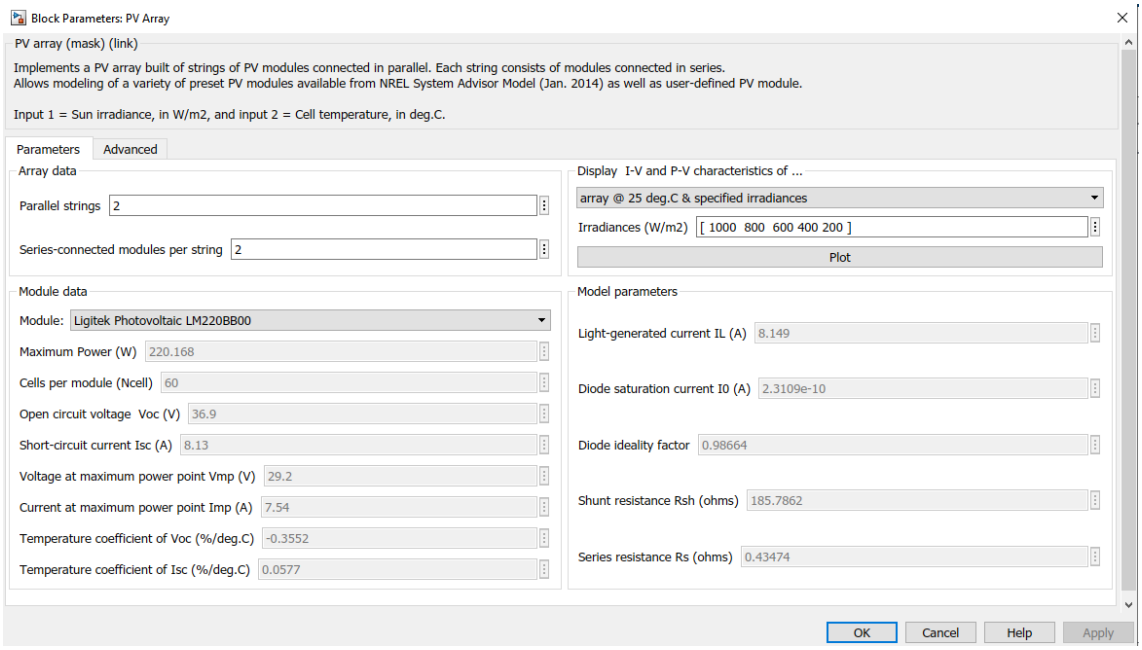

Figura 15: Parámetros del bloque PV array

Se parte del modelo mostrado en la figura 14, pero tras realizar varias pruebas con entradas constantes de temperatura e irradiancia de 25ºC y 1000W/m^2 respectivamente, salía un error en el bloque "PV array". Este error solo se generaba cuando, el "solver" era de paso fijo y en el bloque del panel fotovoltaico estaba seleccionada la opción "Break algebraic loop in internal model" de la pestaña "Advanced", todo ello necesario para poder generar código C del modelo. El error mencionado consistía en que, para un determinado tiempo, la exponencial con la que se calculaba la ecuación linealizada del diodo que constituye el panel fotovoltaico alcanzaba un valor infinito y la simulación se terminaba. A veces este mismo error se producía en otros elementos del modelo, pero el más común era el error en el panel fotovoltaico.

Hablando con los técnicos de soporte de Mathworks, se llegó a la conclusión de que el bloque mencionado tenía un error que los propios técnicos no eran capaces de solucionar y lo actualizarían para futuras versiones. Como no se podía trabajar con el bloque "PV array" debido al fallo producido, se cambió este bloque por el circuito eléctrico de un panel solar fotovoltaico como el mostrado en la figura 7. De esta forma, el modelo resultante es el expuesto en la figura 16. Este modelo final tiene los mismos elementos que el modelo principal, las diferencias son el subsistema que modela el panel fotovoltaico y que no se puede trabajar con la temperatura como parámetro de entrada, debido a que el diodo elegido no tiene el puerto de la temperatura.

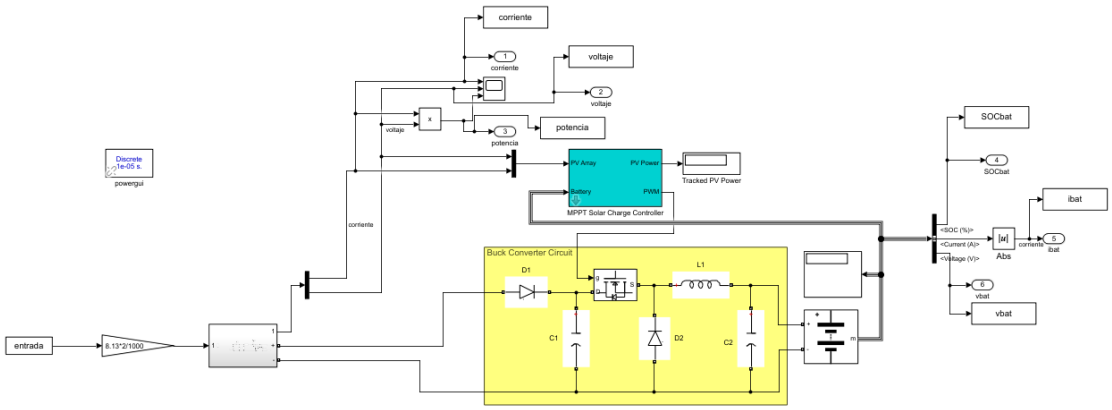

Figura 16: Modelo de la planta solar fotovoltaica completa con el circuito eléctrico del panel solar

Dentro del bloque del panel fotovoltaico se encuentra su circuito eléctrico equivalente (figura17):

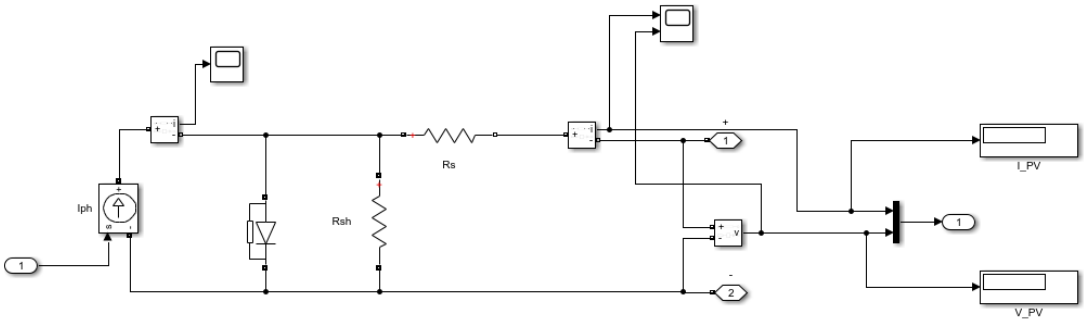

Figura 17: Circuito eléctrico del panel fotovoltaico

Y en el interior del bloque "MPPT Solar Charger Controller" está el diagrama de bloques mostrado en la figura 18.

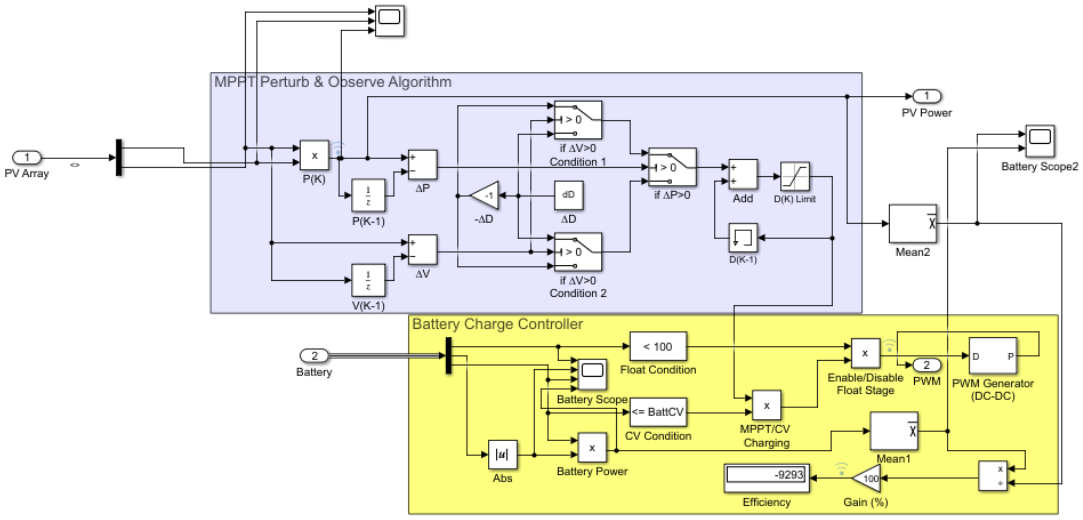

Figura 18: Interior del bloque MPPT Solar Charger Controller

A la vista del modelo elegido, a continuación se desarrolla una descripción de los componentes que forman el modelo final:

- Circuito eléctrico de un panel solar fotovoltaico

Como no se puede trabajar con el bloque PV array, se sustituye por su circuito eléctrico corriespondiente, mostrado en la figura 17. La funcionalidad de este bloque es la misma que la del PV array y para simular el mismo comportamiento en ambos bloques los parámetros tienen que ser iguales.

El valor de entrada del generador de corriente del circuito eléctrico se corresponde con la corriente de cortocircuito (8,13A) multiplicado por 2 debido a que hay dos módulos conectados en paralelo y dividido entre 1000, ya que el parámetro Isc está definido para una irradiancia de 1000W/m^2. Por lo tanto, el generador de corriente recibe la siguiente entrada:

$$\text{Iph} = \frac{8{,}13 * 2}{1000} * \text{irradiancia}$$

La tensión directa del diodo se corresponde con la tensión en circuito abierto (36,9V) multiplicada por 2 puesto que hay dos módulos conectados en serie. La resistencia en paralelo tiene el mismo valor que la resistencia R_{SH} del bloque PV array (185,7862Ω) pero dividida entre 2 por los dos módulos en paralelo. Por último, la resistencia en serie es del mismo valor que la resistencia Rs del bloque PV array (0,43474 Ω), puesto que se multiplica por los dos módulos en serie y se divide por los dos módulos en paralelo.

- Circuito convertidor reductor

El circuito convertidor DC – DC es un tipo de convertidor de potencia que transforma la corriente continua de un nivel de tensión a otro de menor nivel. Su circuito eléctrico está formado por un transistor MOSFET, un diodo, un inductor y un condensador. Su funcionamiento se basa en el análisis de los estados de “ON” (conduce) y “OFF” (no conduce) del transistor.

Estado “ON”, el transistor está conduciendo y la corriente circula desde la fuente de alimentación al condensador. La corriente carga la bobina, la cual se encarga de almacenar la energía y además se alimenta a la carga (batería) con el voltaje obtenido del condensador.

Estado “OFF”, el transistor no conduce, es decir, está en estado de corte por lo que se utiliza la energía almacenada en la bobina para que la corriente circule por el circuito. Esta corriente

permite que el condeasdor siga manteniendo el mismo voltaje a la salida para que pueda continuar alimentando a la carga.

Para conmutar entre ambos estados, dependerá de cuanto dura cada uno con respecto a una frecuencia fija. Por lo que, se usa un circuito de oscilación PWM que dependerá del ciclo de trabajo, tomando como referencia el estado de conducción para la conmutación entre los dos estados [11].

- Bateria

Se encargan de almacenar la energía generada para usarla cuando la producción no sea suficiente.

En las figuras 19 y 20 se pueden apreciar los parámetros generales y los parámetros de descarga de la batería respectivamente. Por una parte, se ha seleccionado una batería de plomo – ácido (las más empleadas en sistemas de generación solar fotovoltaica) con una tensión nominal de 48V, una capacidad nominal de 200Ah (puede suministrar 200A durante una hora hasta descargarse completamente), un estado inicial de carga (SOC) del 50% (la batería inicialmente tiene la mitad de la carga) y un tiempo de respuesta de 30s. Por otra parte, los parámetros de descarga de la batería se han establecido a partir de sus parámetros nominales.

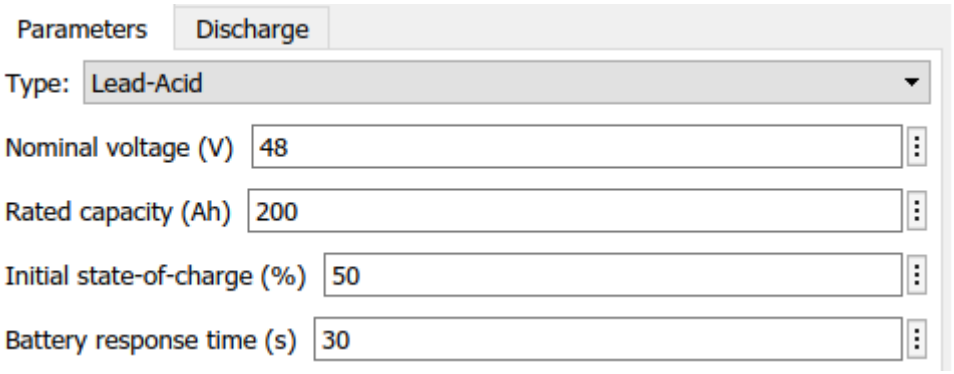

Figura 19: Parámetros generales de la batería

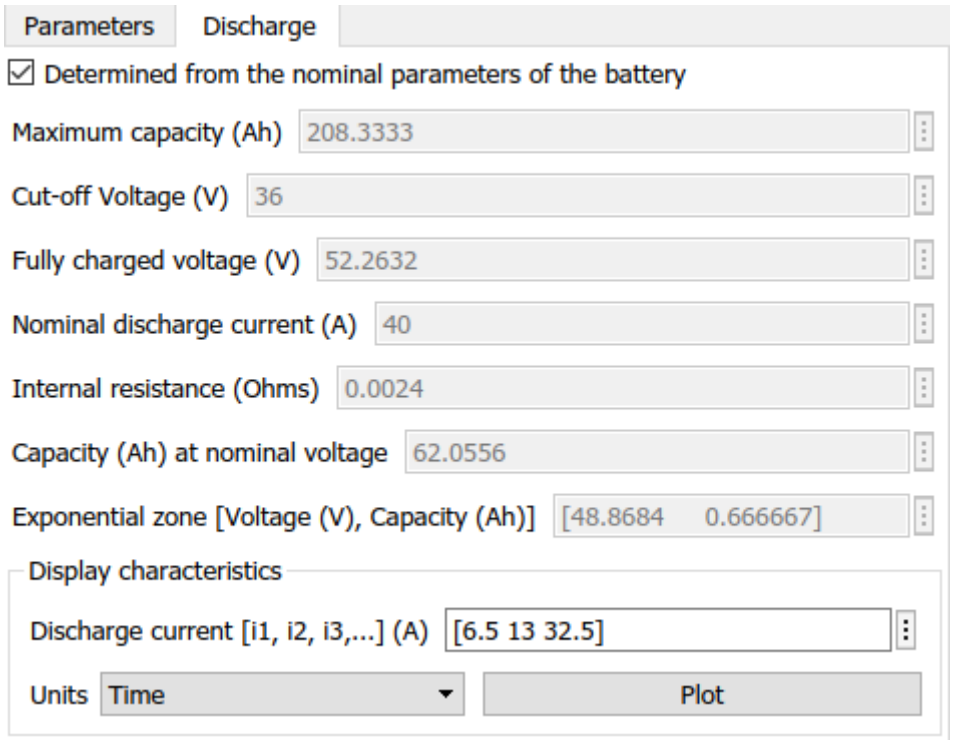

Figura 20: Parámetros de descarga de la batería

- Regulador de carga de la batería

Se encarga de medir el voltaje de la batería para conectarla o desconectarla del módulo fotovoltaico en función de su nivel de carga, consiguiendo así alargar la vida útil de la batería. La desconecta cuando la batería alcanza su carga máxima, comparando el SOC con la carga máxima del 100% y después la vuelve a conectar para que la batería se mantenga cargada gracias a la corriente de flotación.

En este bloque también se puede estudiar la eficiencia de la instalación dividiendo la corriente de la batería entre la potencia de salida del módulo fotovoltaico.

- MPPT con algoritmo P&O

Se hace el seguimiento del punto de máxima potencia para mantener la carga óptima del sistema en función de las entradas del panel solar. La implementación del MPPT se lleva a cabo mediante el algoritmo perturbación y observación. Este algoritmo consiste en observar la variación de voltaje en función de la potencia, de forma que se compara el incremento de tensión con el incremento de potencia. Si ambos incrementos son del mismo signo significa que la tensión de funcionamiento tiene que aumentar para llegar al MPP y si son de diferente signo la tensión de funcionamiento tiene que disminuir. Con este algoritmo se ajusta el ciclo de trabajo, el cual es el encargado de controlar el estado de flotación de la batería.

6.5. Intercambio de datos del gemelo digital con la planta real

Una vez obtenido el modelo en el apartado anterior, hay que establecer un intercambio de datos bidireccional entre el gemelo digital y la planta real para que puedan comunicarse. Como el modelo virtual se implementa en la Raspberry Pi, hay que estudiar los posibles métodos de conexión que existen entre esta placa hardware y el sistema físico. Es necesario que la planta real mande al gemelo digital los datos que obtiene de las entradas (gracias a los sensores conectados a ella) para que los dos modelos tengan los mismo parámetros de entrada. Asimismo, ambos deben

ser capaces de comunicar los datos de salida con el otro para su posterior comparación. Por lo tanto, en este apartado se presentan dos opciones, de todas las posibilidades disponibles, que permiten un intercambio de datos entre dos entidades de forma segura y ordenada.

La primera opción para intercambiar información entre dos equipos es mediante un Socket. Este, permite el intercambio de paquetes de datos, manteniendo su secuencia de envío ordenada, de manera fiable entre dos programas en ejecución, los cuales pueden estar ubicados en diferentes máquinas. El Socket queda definido por dos direcciones IP (una para localizar la computadora local y otra para la remota), un protocolo de transporte y dos números de puerto, los cuales identifican al programa local y al remoto [12]. El funcionamiento del Socket es el mismo que el de un fichero, en él se puede leer y escribir, de forma que la planta real y el gemelo digital depositen los datos necesarios para una buena comunicación.

La segunda opción es la nube, la cual ofrece un potente algoritmo de análisis, con el que se puede realizar el mantenimiento predictivo de la planta real tras depositar los datos del gemelo virtual y el equipo físico. Para realizar la conexión entre ambos equipos se utiliza la herramienta de ThingSpeak. La planta real obtiene la información de varias variables de entrada de los sensores IoT, estos datos se guardan en un campo de ThingSpeak del que posteriormente, el gemelo digital accederá a ellos para adquirirlos como parámetros de entrada en su modelo. Con las salidas se sigue el mismo funcionamiento, la planta real y el modelo virtual envían los datos de salida a ThingSpeak donde se almacenan en campos diferentes, de manera que se puedan comparar los resultados obtenidos de ambos sistemas. El problema de ThingSpeak es que no actualiza los datos en tiempo real, sino que tarda 20 s en actualizarlos, por lo que para aplicaciones que se ejecutasen muy rápido esta herramienta no sería útil. En el caso de la planta solar, la actualización de los datos no sería un problema, puesto que la Raspberry Pi lee la información de entrada más rápido que lo que tarda en actualizarse la nube, obtendría el mismo dato de entrada repetido varias veces, y por consiguiente el mismo dato de salida repetido, antes de que se actualice pero no afectaría al modelo. Otro de los problemas que presenta esta herramienta es su difícil implementación en código tanto en Simulink como en PV_LIB. En Simulink porque el bloque de ThingSpeak que obtiene los valores de entrada de la nube, presenta varias complicaciones a la hora de compilar el código impidiendo que se pueda ejecutar ese modelo. Por lo que, si se quieren obtener los datos de la nube, una vez generado el código habría que leer la información de ThingSpeak con la librería “curl”, la cual permite obtener la información que aparece en esa página web. Y en PV_LIB, el uso de la función de ThingSpeak no es compatible a la hora de generar el código, por lo que tampoco se puede utilizar.

En este proyecto, para establecer la comunicación entre el gemelo digital y la planta real, se emplea ThingSpeak. Para establecer en el modelo virtual, las mismas condiciones de entrada que recibe el sistema físico. En cuanto a la variables de salida, como ThingSpeak solo deja enviar un dato cada 20 s, se utiliza un archivo de texto para guardar toda la información obtenida del modelo. Al guardar los datos en un archivo de texto, desde Matlab se puede acceder a ellos fácilmente y los puede comparar con los resultados que se obtienen de la planta real.

6.6. Herramientas software

El desarrollo del gemelo digital se ha llevado a cabo mediante dos métodos diferentes: Simulink y PV_LIB, ya que con el primero no se puede modelar el efecto de la temperatura y el segundo no se puede implementar en la placa hardware. A parte de estas Toolboxes de Matlab, también se ha necesitado la ayuda de otras herramientas como ThingSpeak para obtener los datos de entrada de la nube, PVGIS para obtener los datos de irradiancia reales de un día concreto y VNC viewer para conectar el ordenador a la Raspberry Pi de forma remota.

6.6.1. Matlab

Matlab es una plataforma de programación y cómputo numérico que cuenta con un entorno de desarrollo integrado y un lenguaje de programación propio basado en matrices, el lenguaje M. Entre sus prestaciones cabe destacar el análisis de datos, representación de datos y funciones mediante gráficas, desarrollo e implementación de algoritmos, creación de aplicaciones web y de escritorio, capacidad de usar Matlab con otros lenguajes, cálculo en la nube y conexión entre Matlab y otros equipos hardware [13]. Asimismo, este software, usado principalmente en un ámbito académico y de investigación, cuenta con dos herramientas adicionales, Simulink (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario).

6.6.2. Simulink

Simulink es un entorno de programación visual basado en diagramas de bloques proveniente del entorno de programación de Matlab. Esta herramienta se usa para diseñar sistemas basados en modelos multidominio, diseñar y simular antes de la implementación hardware y desplegar sin necesidad de escribir código. Simulink es un entorno de programación de más alto nivel de abstracción que Matlab y los archivos que genera tienen una extensión .mdl, a diferencia de Matlab que cuenta con una extensión .m [14].

6.6.3. PV_LIB

PV_LIB es una Toolbox formada por un grupo de funciones para simular la energía generada en los sistemas fotovoltaicos. Existen dos versiones diferentes para esta Toolbox, la de Matlab y la de Python, sin embargo, este trabajo se centrará en la Toolbox PV_LIB para Matlab. Originalmente, se desarrolló en “Sandia National Laboratories”, pero con el tiempo se ha convertido en un software de código abierto. Esta Toolbox permite calcular la irradiación y posición solar, descomposición de la irradiancia y transposición al plano de la matriz, suciedad y sombreado, temperatura de la celda, obtención de la potencia a través de la irradiancia, pérdidas por desajuste en la resistencia y por desajuste eléctrico de CC, seguimiento del punto de máxima potencia, eficiencia del inversor, pérdidas de CA y degradación a largo plazo [15].

6.6.4. ThingSpeak

Fue fundada en 2010 por ioBrige como un servicio de soporte para aplicaciones de IoT. ThingSpeak es un servicio de plataforma de análisis de IoT que consiste en agregar, visualizar y analizar flujos de datos en tiempo real en la nube. Se pueden enviar datos a ThingSpeak desde dispositivos habilitados para internet, visualizando los datos en vivo y enviando alertas [16]. Más concretamente, con ThingSpeak se pueden recoger los datos de los sensores IoT y enviarlos de forma privada a la nube, analizar y visualizar los datos con la herramienta de Matlab y activar reacciones en función de los datos recogidos para enviar notificaciones o alertas usando aplicaciones como Twitter y Twilio.

ThingSpeak está formado por canales públicos y canales privados, para estos últimos, si se accede o trabaja con ellos, se necesita la contraseña del canal. Para acceder a un canal público basta con el ID del canal y especificar el número del campo con el que se va a trabajar. En cambio, para acceder a un canal privado se necesita el ID del canal, la clave de lectura API y el número del campo de trabajo. Dentro de un canal hay varios campos en los que se guardan diferentes variables como temperatura, humedad, irradiancia, etc. Por eso, a la hora de trabajar con ThingSpeak, hay que especificar el campo que se requiere en cada momento.

6.6.5. Sistema de Información Geográfica Fotovoltaica (PVGIS)

PVGIS es un centro científico de la unión europea que proporciona información sobre la radiación solar y el rendimiento del sistema fotovoltaico para toda Europa y África y gran parte de Asia y América [17]. Esta calculadora cuenta con siete menús, de los cuales tres de ellos permiten calcular la generación de energía, otros tres permiten calcular la irradiancia de entrada y el último obtiene los datos de un año meteorológico típico (TMY). Los tres menús que calculan la generación de energía pueden ser de sistemas conectados a la red, sistemas autónomos o de sistemas PV de seguimiento. Los tres menús que obtienen la irradiancia de entrada del panel se pueden agrupar en datos mensuales, datos diarios y datos horarios. Y el último menú logra, para diferentes variables, los datos meteorológicos en un año según el periodo de tiempo indicado. Para establecer la ubicación del lugar de estudio, cuenta con un mapa interactivo en el que se indica la ciudad o se puede introducir directamente las coordenadas de la posición deseadas. Los resultados logrados con las simulaciones se pueden obtener en archivo CSV, PDF o visualizarse en archivos HTML.

Para realizar la comparativa de los modelos empleados en el desarrollo del gemelo digital, en vez de trabajar con ThingSpeak (puesto que ocasiona algunos problemas en su implementación) se utiliza la entrada de irradiancia obtenida de PVGIS. Se descargan los datos diarios de la irradiancia para el mes de julio, estableciendo los parámetros de entrada como los expuestos en la figura 21, en un archivo CSV. En este archivo se obtienen los valores de irradiancia total, directa y difusa, pero solamente se trabaja con el valor de la irradiancia total.

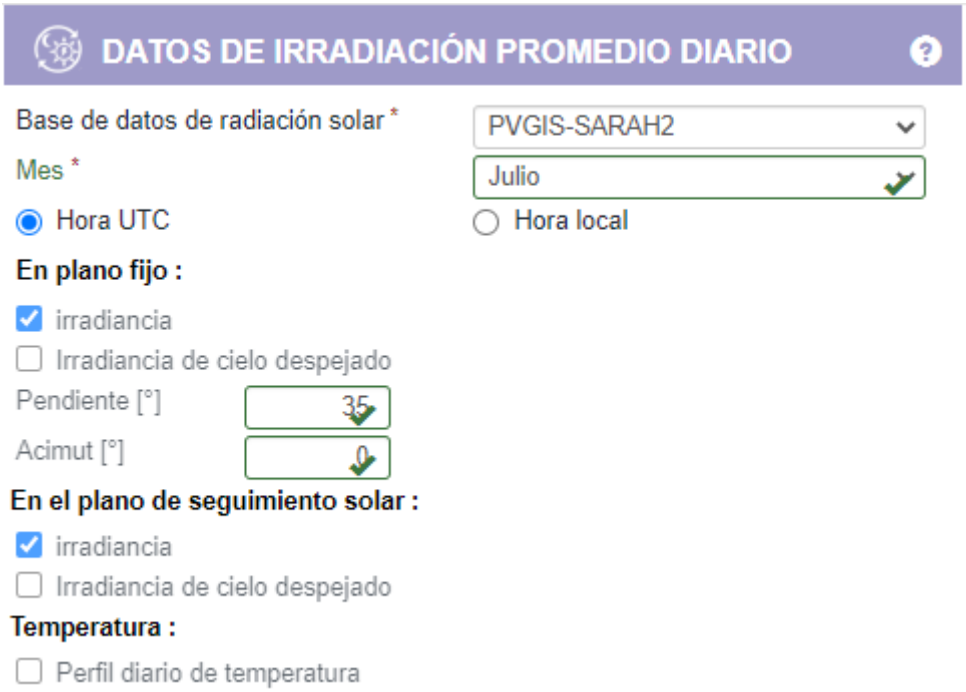

Figura 21: Datos de irradiación promedio diarios

6.6.6. VNC viewer

VNC viewer (Virtual Network Computing) es un software de código libre basado en una estructura cliente – servidor, en el que el ordenador cliente puede ver la pantalla y controlar e interactuar con el equipo del ordenador servidor remotamente. VNC no tiene en cuenta el sistema operativo del ordenador servidor con respecto al ordenador cliente, para establecer la comunicación se necesita cualquier sistema operativo que admita VNC. Este software fue creado

en Reino Unido, específicamente en “AT&T Olivetti Research Laboratory” basado en el protocolo RFB [18].

VNC, aunque no es imprescindible para el desarrollo de este proyecto si es de utilidad. Pues, se puede controlar la pantalla de la Raspberry Pi remotamente en el ordenador de trabajo sin la necesidad de estar con dos pantallas a la vez (una para el PC de trabajo y otra para el monitor que se conecta a la placa hardware) o tener que trabajar desde casa. Este software sirve de ayuda si no se dispone de monitor, teclado y ratón para poder ver y controlar la Raspberry Pi.

Para poder trabajar con VNC viewer, en la pestaña de Interfaces, dentro de la configuración de la Raspberry Pi tiene que estar habilitada la opción VNC. Con esta opción habilitada, aparecerá el símbolo de VNC en la barra de tareas. Para añadir la Raspberry Pi, basta con crearse una cuenta en la página de VNC y asignarle un nombre a esa Raspberry Pi con la dirección IP obtenida, puesto que en esta aplicación se pueden controlar varios equipos con diferentes direcciones IP. Finalmente, en el ordenador servidor, una vez que se descarga la aplicación, hay que escribir la dirección IP para agregar el dispositivo de la Raspberry Pi.

6.7. Herramientas hardware

Para implementar el gemelo digital hay que tener en cuenta una serie de condiciones que se deben cumplir. El equipo hardware donde se lleva a cabo la aplicación del gemelo digital tiene que ser de bajo coste y alta fiabilidad para asegurar una correcta transmisión de los datos. Se necesita establecer una comunicación con la nube para obtener los parámetros de entrada del modelo y, en caso de que haya más equipos, comunicarse entre ellos. Por último, el dispositivo debe contar con la potencia suficiente para poder ejecutar el gemelo digital en tiempo real, ya que es una de sus principales características.

Dentro de las posibilidades que presenta el mercado, para este TFG se ha elegido la Raspberry Pi 4 model B, puesto que este modelo era compatible con la versión 2022a de Matlab y Simulink.

6.7.1. Raspberry Pi

La raspberry es un ordenador de bajo costo, poca potencia y tamaño reducido. Su objetivo no es tener un procesador muy potente, sino que es un pequeño ordenador que puede estar encendido 24 horas al día ya que casi no consume. La placa hardware fue diseñada por Raspberry Pi Foundation y se creó para dotar de ordenadores de baja potencia muy económicos a niños para que aprendiesen informática. Al ser tan baratas tuvieron mucho éxito y se empezaron a usar para otros objetivos diferentes a los que tenían en mente en la Raspberry Pi Foundation [19]. Hay varios modelos, de los cuales la Raspberry Pi 1 model A y model B ya están descatalogados, mientras que en este trabajo se utiliza la Raspberry Pi 4 model B. Todos estos ordenadores de placa simple cuentan con un procesador Broadcom que varía en función del modelo.

Este ordenador de placa reducida necesita unos componentes extra para su funcionamiento. Necesita una tarjeta microSD para cargar el sistema operativo con el que funcionará la Raspberry (en concreto, en este proyecto se trabaja con una tarjeta microSD de 32GB). Su sistema operativo oficial es el Raspberry Pi OS, el cual es una versión de Debian, aunque también admite usar otros sistemas operativos como el Windows 10. También se necesita una fuente de alimentación externa para la Raspberry Pi, el cual debe tener como mínimo 2A de corriente.

La Raspberry Pi es bastante interesante para los siguientes proyectos: se puede utilizar como un mini ordenador, aunque hay que tener en cuenta que es de baja potencia, un centro multimedia, ya que tiene un puerto HDMI que se puede conectar a la televisión para ver vídeos que estén guardados o que se carguen a través de los puertos USB, puede utilizarse como un servidor

privado en la nube seguro con almacenamiento casi ilimitado, se puede ejecutar un emulador de consola y puede emplearse para proyectos de domótica y de electrónica. Para estos últimos proyectos, la Raspberry Pi es muy potente porque tiene unas entradas/salidas digitales que pueden conectarse a sensores, cámaras, etc.

6.7.1.1. Raspberry Pi 4 modelo B

Este modelo está constituido por un procesador Broadcom BCM2711 Cortex – A72 nuevo tres veces más eficiente que el anterior, de 4 núcleos de 64 bits SoC de alto rendimiento. Evolucionan los puertos HDMI a dos puertos microHDMI a través de los cuales se consigue una pantalla dual de hasta 4K de resolución, decodificación de video por hardware de hasta 4Kp60, doble LAN inalámbrica de banda 2.4/5.0 GHz, Bluetooth 5.0, Gigabit Ethernet, capacidad PoE y es la primera Raspberry Pi en la que aparecen puestos USB 3.0 [20]. En la figura 22 se pueden visualizar todos los componentes que forman la placa hardware. Y en la figura 23 se pueden observar las dimensiones de la placa, de la cual se confirma que la Raspberry Pi 4 model B es un dispositivo de tamaño reducido puesto que es de 85x56mm.

Figura 22: Raspberry Pi 4 model B

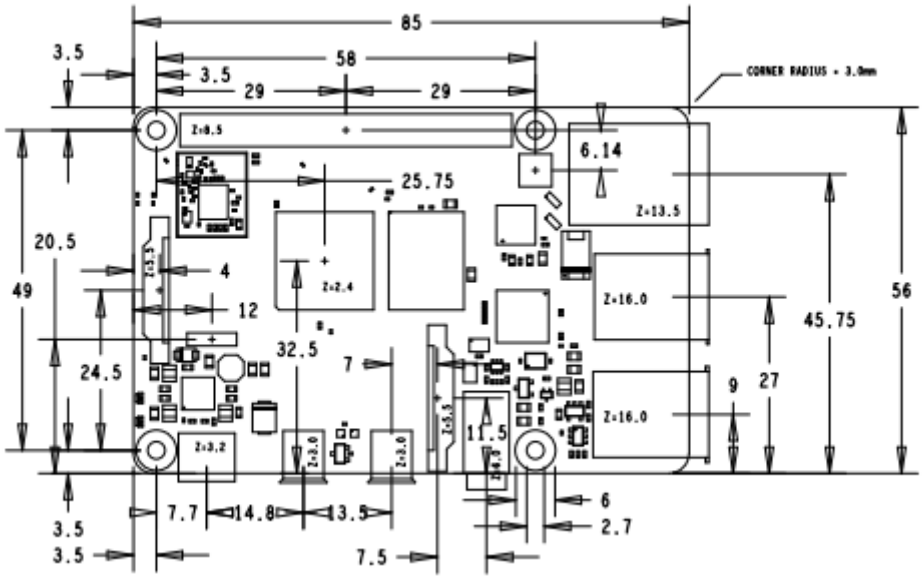

Figura 23: Dimensiones mecánicas Raspberry Pi 4 model B

6.8. Comunicación de la Raspberry Pi con otros entornos

La implementación del gemelo digital se lleva a cabo en la Raspberry Pi ya que cumple con los requisitos de implementación hardware, pero el desarrollo del modelo de la planta solar se lleva a cabo en dos entornos de programación: Simulink y PV_LIB.

En cada entorno, el modelo virtual se obtiene mediante diferentes métodos. En Simulink se logra mediante un diagrama de bloques, el cual se convertirá a código C gracias a una de las aplicaciones de esta herramienta. En PV_LIB, se desarrolla la entidad virtual a través de las funciones que presenta esta Toolbox para modelar la energía de los sistemas fotovoltaicos. Como ambas Toolboxes son de Matlab, se necesitan instalar dos paquetes de Soporte para la Raspberry Pi, uno para Matlab y otro para Simulink.

6.8.1. Comunicación entre la Raspberry Pi y Matlab

Para poder trabajar con la Raspberry Pi en el entorno de Matlab lo primero que hay que hacer es instalar el paquete de soporte de Matlab y Simulink para la Raspberry Pi, es decir, los paquetes "Simulink Support Package for Raspberry Pi Hardware" y "MATLAB Support Package for Raspberry Pi Hardware". También, es necesario que en la configuración de la placa hardware, en la pestaña de "interfaces" esté habilitada la opción "SSH" para que se pueda establecer la comunicación.

6.8.2. Paquetes de soporte de Simulink y Matlab para la Raspberry Pi

Los paquetes de soporte se encuentran dentro de la pestaña "HOME" de Matlab, en "Add – ons" hay que seleccionar "Get Hardware Support Package". Para poder trabajar con estos paquetes de soporte hay que seguir dos pasos, primero la instalación y segundo la configuración.

- Primer paso: instalación

Una vez seleccionado los paquetes de soporte deseados hay que instalarlos.

- Segundo paso: <u>configuración</u>

Para poder configurarlos, hay que seleccionar el botón de ajustes que aparece al lado de cada paquete instalado. Lo primero que pide es establecer el modelo de trabajo, en este caso es la Raspberry Pi 4 model B (figura 24).

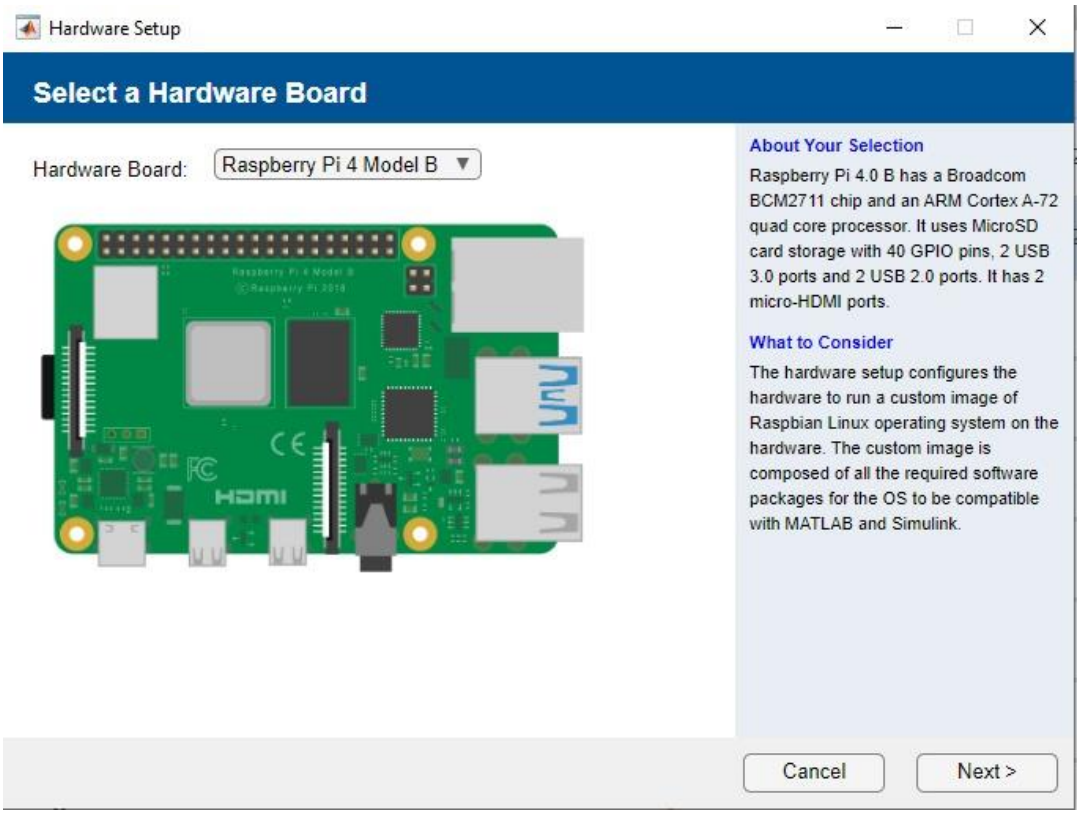

Figura 24: Selección de la placa Hardware

A continuación, hay que elegir el sistema operativo que se va a cargar en la tarjeta microSD. Para ello, se elige la opción “Setup hardware with MathWorks Raspbian image” y descargar el archivo “MathWorks Raspbian image” (figura 25).

Name	Version	Details
MathWorks Raspbian image (mathworks_raspbian_R22.1.0.zip)	22.1.0	Download

Figura 25: Descarga del sistema operativo

Cuando ya se ha descargado el sistema operativo que se va a cargar en la tarjeta microSD hay que validarlo. Para esto, en la siguiente ventaja se selecciona el archivo que se ha descargado y se pulsa el botón de “validate”. Para saber si están bien, tiene que mostrarse por pantalla algo similar a la figura 26.

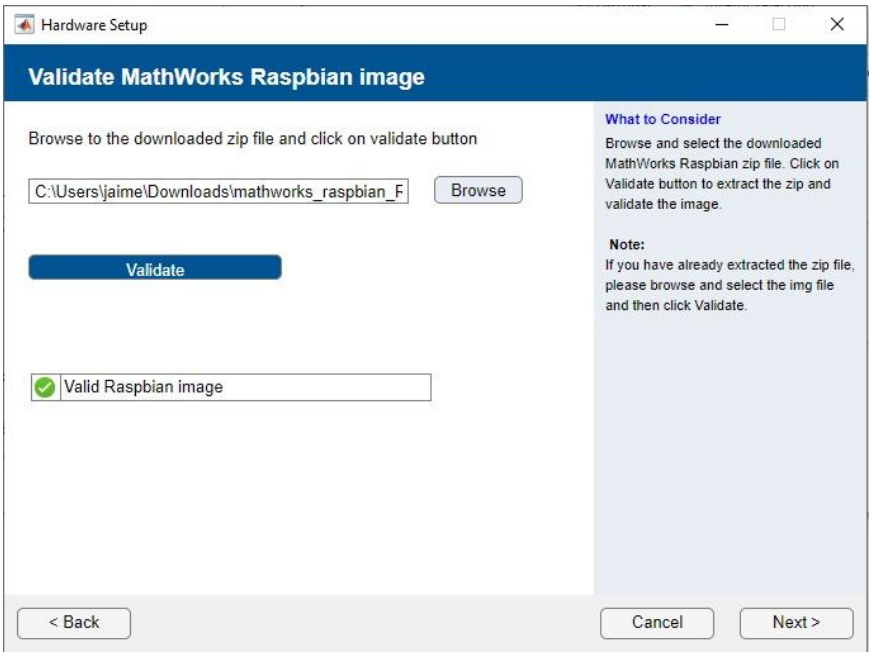

Figura 26: Validación del OS Raspbian

Para establecer el tipo de conexión entre la Raspberry Pi y el PC en el que esté instalado Matlab hay varias posibilidades, todas ellas expuestas en la figura 27, en este caso se trabaja con una conexión a través del WIFI. Por lo tanto, se selecciona la segunda opción "Connet to Wireless network".

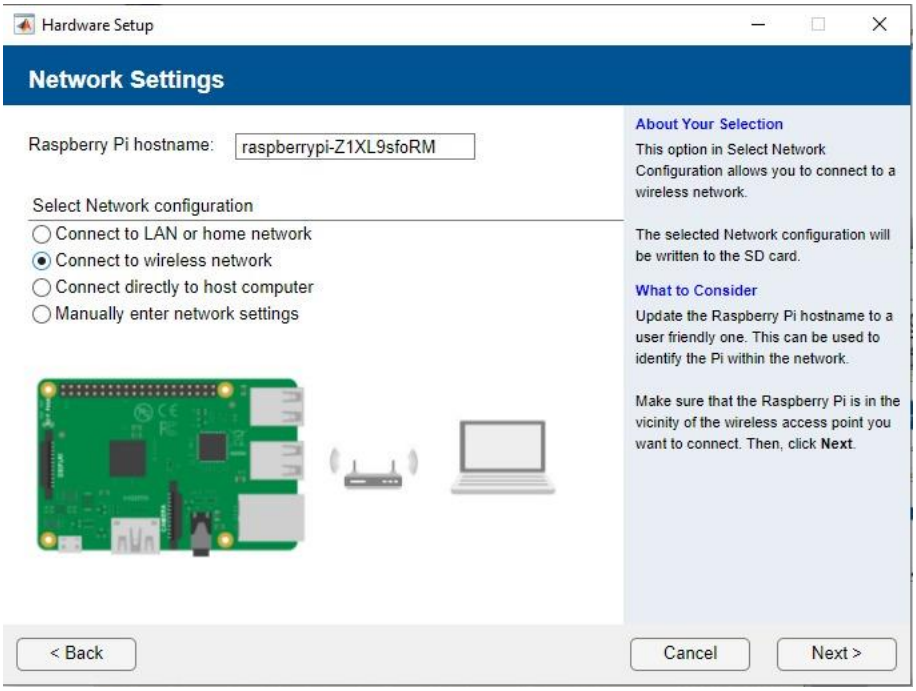

Figura 27: Configuración de la red

Para establecer la red en la que se va a trabajar, en la próxima pestaña hay que introducir el nombre de la red WIFI y su contraseña. En este momento se le asigna una dirección IP a la Raspberry Pi, con la opción de establecerla automática o manualmente. En este proyecto se ha elegido la opción por defecto, es decir, automáticamente.

Con todas las opciones ya establecidas solo queda cargar el sistema operativo en la tarjeta microSD. De manera que se introduce la tarjeta microSD en el lector de tarjetas del ordenador y

automáticamente en la casilla de “Drive” aparecerá el nombre de la tarjeta microSD. Cuando ya se ha cargado completamente el sistema operativo hay que introducir la tarjeta microSD en la Raspberry Pi.

Una vez realizado todos los pasos, tiene que aparecer una pestaña final como la mostrada en la figura 28 donde se confirma que la configuración hardware se ha establecido correctamente y en la que se puede observar la dirección IP asignada a la Raspberry Pi.

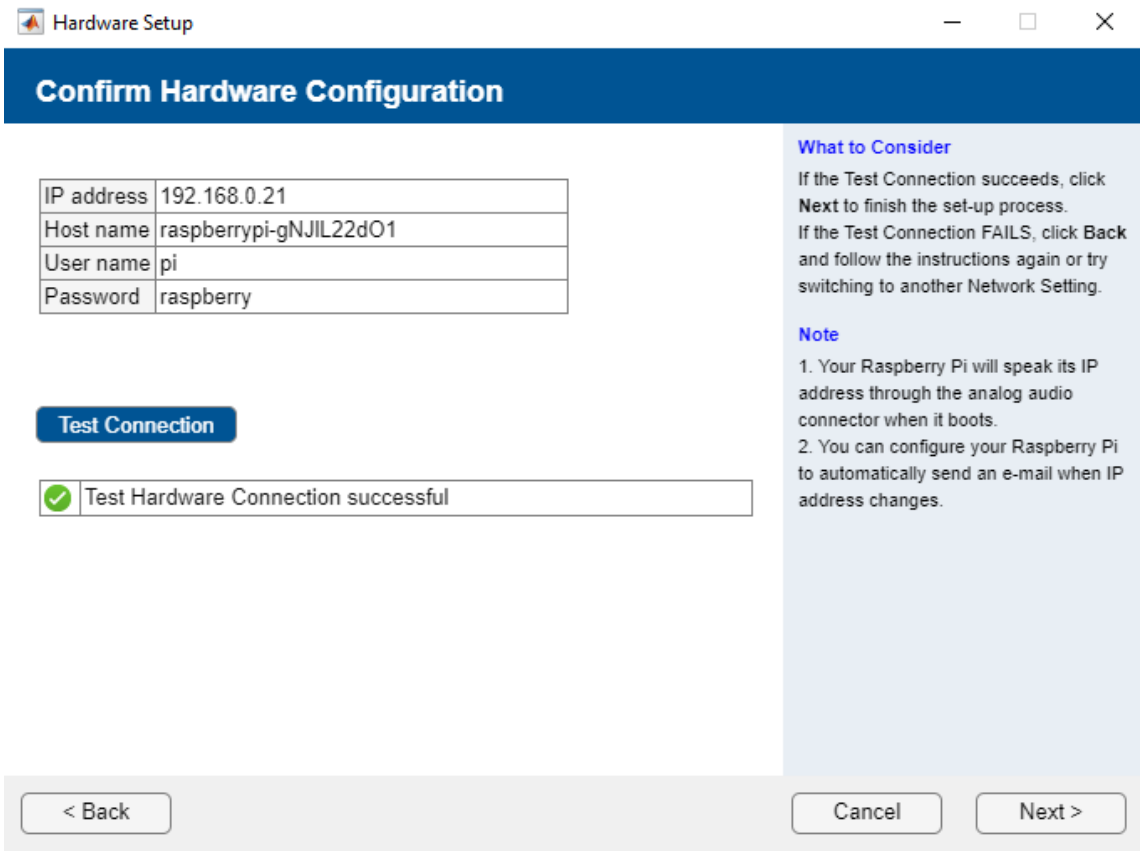

Figura 28: Confirmación de la configuración hardware

7. Métodos para la construcción del gemelo digital

Como ya se ha mencionado anteriormente, se va a construir el gemelo digital a través de dos procedimientos diferentes (Simulink y PV_LIB), cada uno con sus ventajas e inconvenientes.

7.1. Simulink

El modelo virtual del gemelo digital que se ha desarrollado en este entorno de programación es el mostrado en la figura 15.

El inconveniente de este modelo es que no se ha podido modelar el efecto de la temperatura en el circuito eléctrico del panel fotovoltaico, puesto que el diodo seleccionado no daba la opción de mostrar el puerto de la temperatura, así que solo se trabajará con la irradiancia como parámetro de entrada. Por este motivo, no es una representación realista de la planta solar que se encuentra en el Edificio Politécnico, ya que la temperatura es un parámetro importante que condiciona la potencia de salida del panel fotovoltaico. Sin embargo, la ventaja de este modelo es que cuenta con los mismo componentes que el modelo físico, así que, por este lado, sí que sería una representación fiel de la planta solar fotovoltaica real.

Se estudió también el caso de modelar el efecto de la temperatura con un diodo de la librería "Simscape", resultado un circuito eléctrico del módulo fotovoltaico como el expuesto en la figura 29.

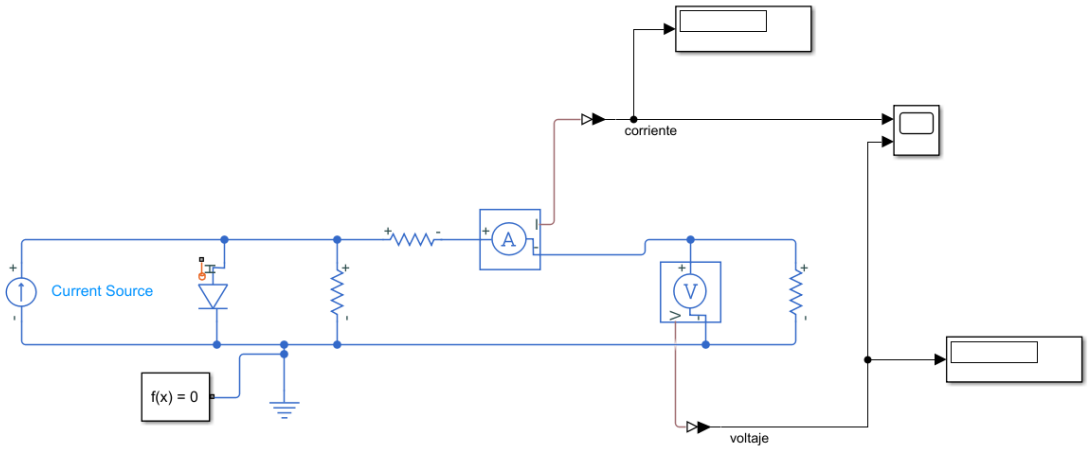

Figura 29: Circuito eléctrico del panel fotovoltaico con el efecto de la temperatura

Sin embargo, este modelo, al usar elementos de la librería "foundation", a la hora de ejecutar el código C presentaba numerosos problemas y no se podía compilar. Por lo que, se descartó este modelo y no se pudo tener en cuenta la temperatura como parámetro de entrada.

Cabe destacar la difícil implementación de los modelos a través del código C creado por Simulink debido a que presentaba ciertos problemas para compilarlo. Estos problemas se presentaban en algunas librerías (como la librería foundation) y bloques de Simulink (como el bloque de ThingSpeak para la Raspberry Pi).

7.1.1. Conexión entre simulink y la Raspberry Pi

Para enviar los modelos creados desde simulink a la Raspberry Pi, se generaba un código C que posteriormente habría que modificar para que guarde las variables de estudio en un archivo de

texto. Para crear este código directamente desde Simulink, hay dos opciones. En la primera se genera el código con la ayuda de una aplicación que indica los parámetros que hay que establecer paso a paso y en la segunda se genera el código directamente en la Raspberry Pi.

Generar código mediante la aplicación

En la pestaña de "APPS" se elige la aplicación "Embedded Coder" y se crea código con la ayuda del "Quick Start" que ofrece Simulink. Esta opción, va paso a paso mostrando la configuración con la que se obtendrá el código final en C o C++, en este caso se trabaja con código C.

Pasos en la configuración del código:

- Pestaña "System": se selecciona el modelo o el subsistema para la generación de código. En este caso siempre se trabaja con la opción de crear código del modelo completo. Al principio se partía de modelos más simples y cuando ya se tenía el método aprendido se generaba código de los modelos finales.
- Pestaña "Output" (figura 30): se establece la salida y la cantidad de instancias que se necesitan para la generación de código. Con todos los modelos que se ha trabajado, se elige la salida de código C y las instancias "Single".

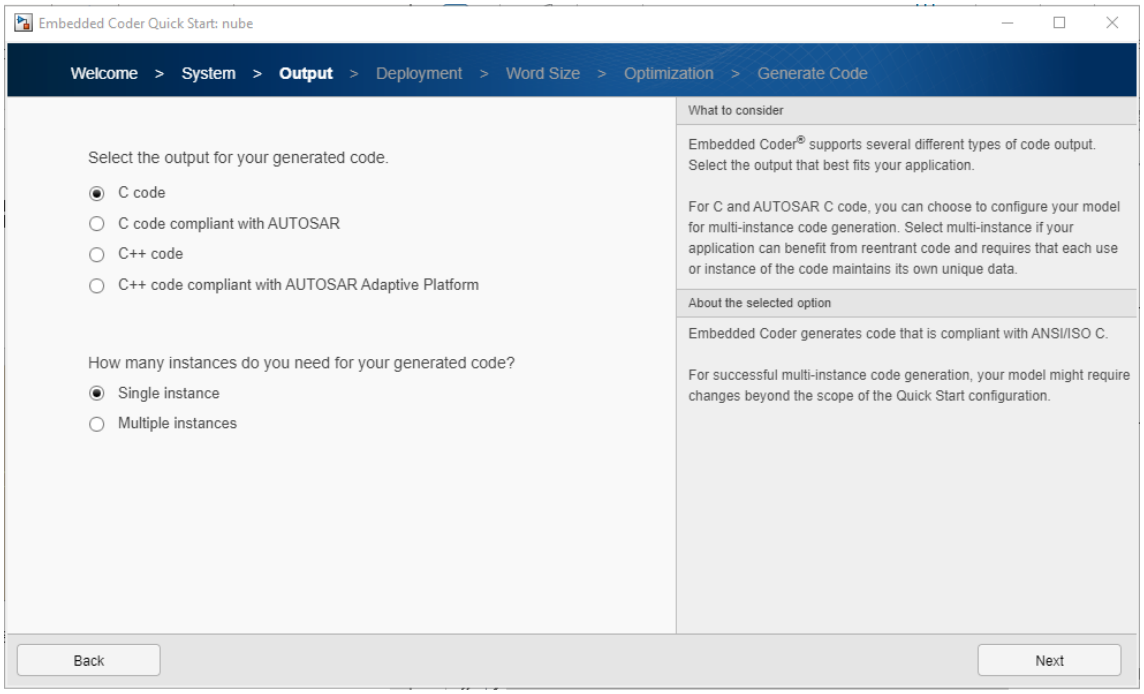

Figura 30: Salidas en la generación de código

- Pestaña "Deployment": calcula cuantas frecuencias de muestreo hay en el modelo, si el sistema diseñado contiene bloques continuos, si se configuró el sistema para llamadas de funciones de exportación y si el sistema contiene modelos referenciados. Para el modelo de trabajo, el despliegue del análisis se puede observar en la figura 31.

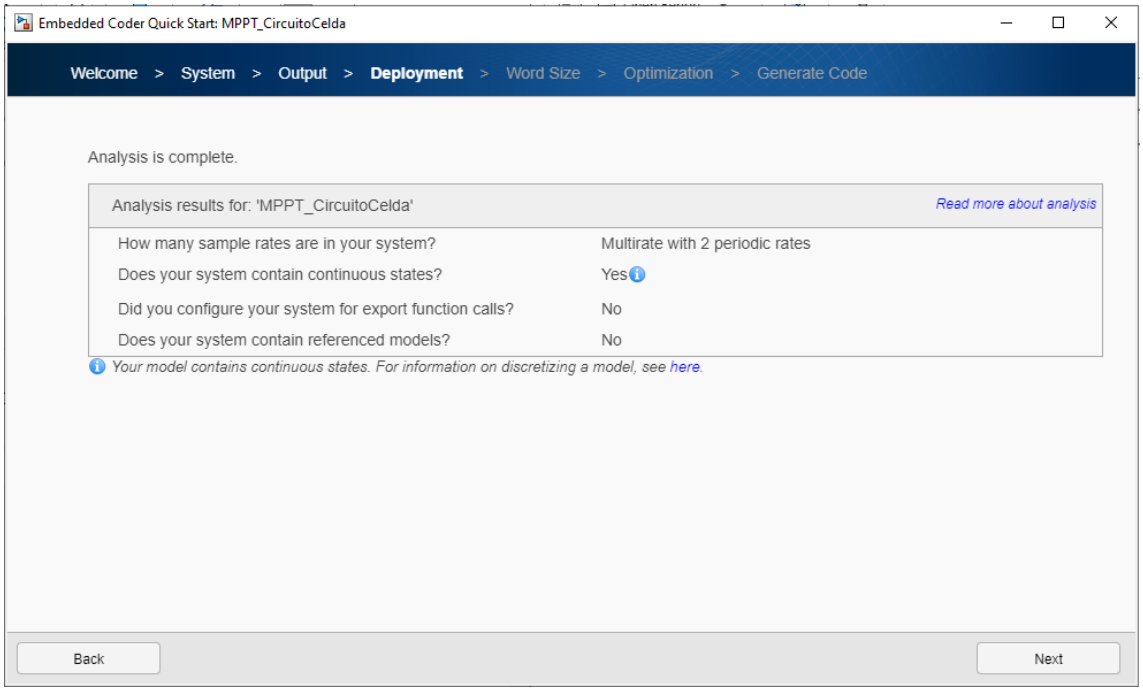

Figura 31: Análisis del modelo

Después de calcular el análisis obtenido, como el modelo de trabajo cuenta con dos frecuencias de muestreo, hay que elegir como se formará el código para la ejecución multitarea. Se elige la opción "Single rate" de forma que se crea una función de punto de entrada única para que se pueda programar como una única tarea a diferencia de la opción "Multitasking rate monotonic" en la que se genera múltiples funciones de punto de entrada que se ejecutan como varias tareas.

- Pestaña "Word Size" (figura 32): se selecciona el tipo de procesador de la placa hardware, en este caso, hay que establecer en el proveedor de dispositivos "ARM Compatible" y en el tipo de dispositivos "ARM Cortex – A", ya que la Raspberry Pi 4 tiene un procesador ARM Cortex – A72.

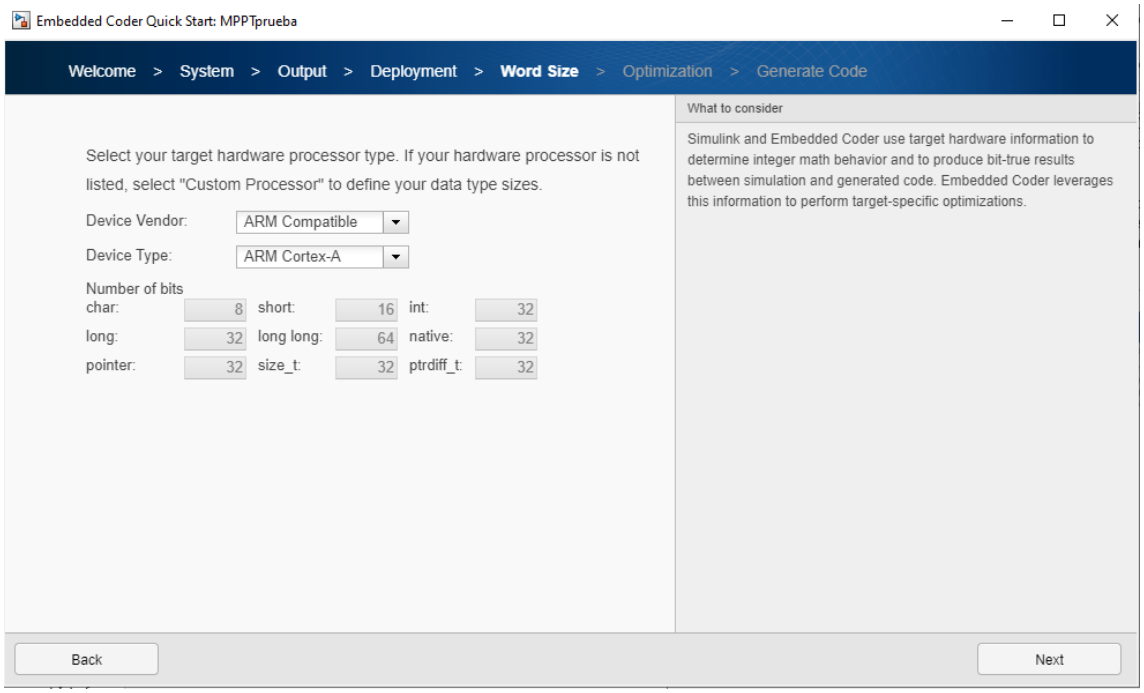

Figura 32: Tipo de procesador de la placa hardware

- Pestaña “Optimization” (figura 33): se selecciona el objetivo más importante en la generación de código. Hay dos opciones, ejecución eficiente o RAM eficiente, con los modelos trabajados se ha priorizado la opción de velocidad de ejecución.

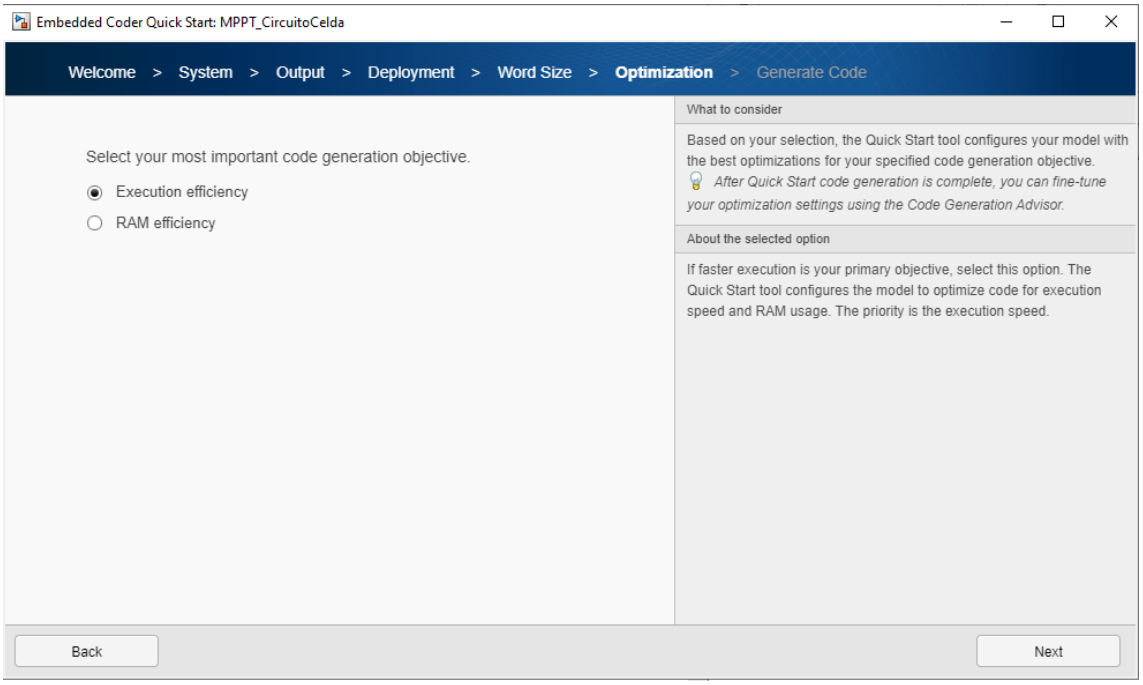

Figura 33: Optimización en la generación de código

- Pestaña “Code Generated”: se muestran los cambios seleccionados en la generación del código, es decir, los valores nuevos y antiguos de cada parámetro modificado y la categoría a la que pertenecen.

Generar código directamente en la Raspberry Pi

Inicialmente, en la pestaña “Hardware Implementation” de “Model Settings”, hay que establecer en “Hardware board” la placa de trabajo, osea la Raspberry Pi. De esta manera, se complentan los campos del tipo de procesador hardware correspondientes a la Raspberry Pi con las mismas características que las mostradas en la figura 32. También se rellenan automáticamente la dirección IP de la Raspberry Pi, su usuario y su contraseña gracias a la configuración establecida en los paquetes de soporte (figura 34).

Una vez que se ha elegido la placa hardware de trabajo, en la nueva pestaña “HARDWARE” que se crea, tiene que estar seleccionado la opción “Run on board” para que permita al ordenador de trabajo comunicarse con la Raspberry Pi. Finalmente, para crear el código hay que optar por el despliegue “build”. Con el cual, aparecerá directamente en el directorio de trabajo de la Rasoberry Pi el código generado.

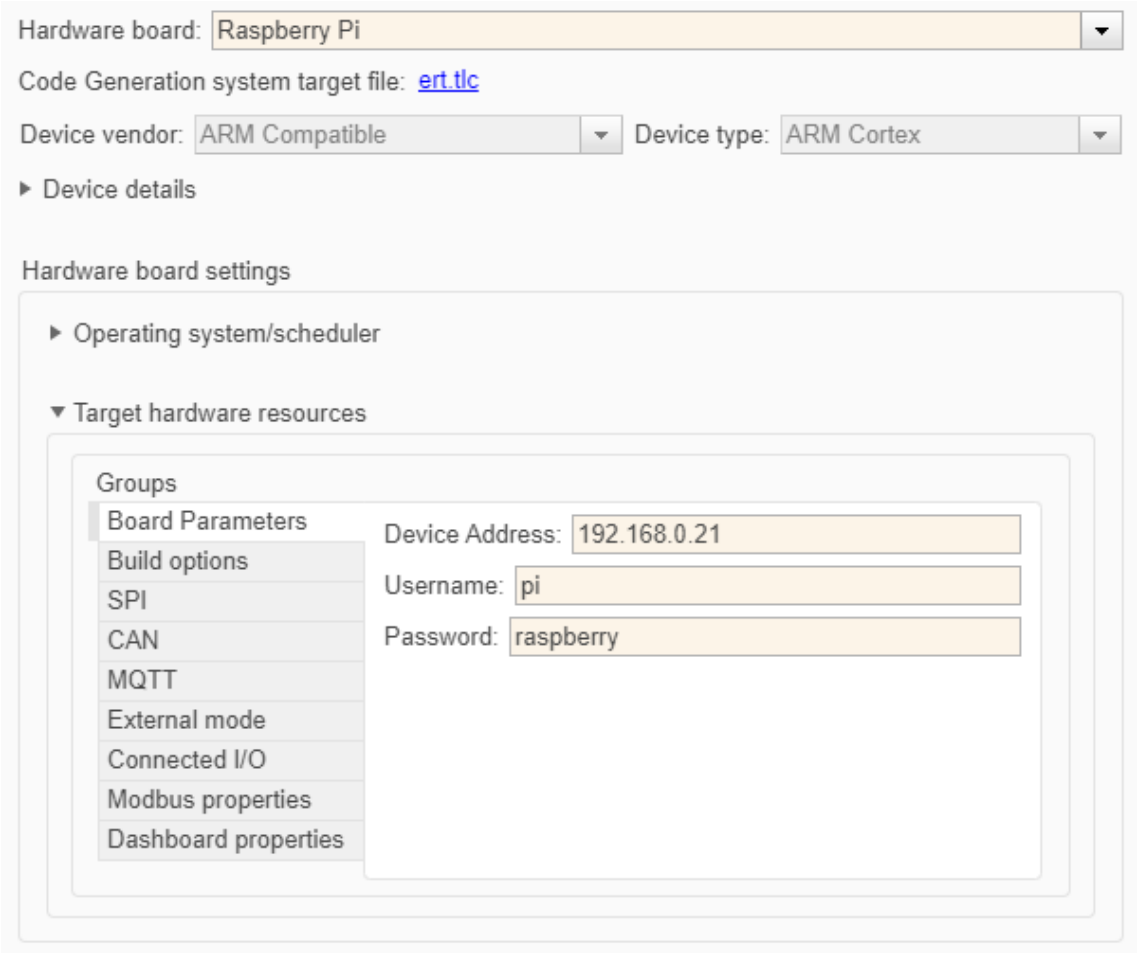

Figura 34: Configuración hardware en Simulink

Con ambos métodos se crea una carpeta que contiene varios archivos .h y .c entre los que se encuentra el "main". Este archivo, es en el que se realizan la modificaciones necesarias para que guarde en un archivo de texto los resultados de los parámetros de salida como la potencia, tensión y corriente del panel fotovoltaico; la corriente, tensión y SOC en la batería y el tiempo que se tarda en llevar a cabo esa operación para saber si se puede considerar en tiempo real o no. Cabe destacar, que para poder modificar las entradas y salidas desde el código, hay que añadir unos bloques de entrada "Inport" y unos bloques de salida "Outport" en Simulink.

Cuando ya se han realizado las modificaciones, hay que depurar el código para ver que no haya errores en el mismo. Paralelamente, hay que añadir los archivos "solver.h" y "common.h" en la carpeta obtenida al generar código en Simulink para que no de errores al depurar el archivo main. Una vez que todo esta correcto, desde el terminal de Linux de la Raspberry Pi, para ejecutar el código, hay que situarse en el directorio en el que se encuentra la carpeta de archivos. Situado en ese directorio hay que crear un archivo ejecutable , el cual se va a encarga de llevar a cabo la aplicación. Para crear el ejecutable, es ncesario añadir al principio de la sentencia el compildor gcc junto con los archivos .c que se han creado en la generación de código del modelo y al final de la sentencia el nombre del archivo ejecutable con el flag "lm" (de la libreía math). Una vez obtenido el archivo ejecutable "ejecutable", ya se puede poner en marcha la aplicación. En la figura 35 se enseñan todas las sentencias y los archivos empleados para la ejecución de la aplicación.

Figura 35: Comandos para ejecutar en la Raspberry Pi el modelo de Simulink

7.2. PV_LIB

El modelo creado con PV_LIB está formado por un panel solar fotovoltaico, ya que son los componentes que modela esta Toolbox.

Pasos y funciones para crear un modelo en PV_LIB:

En primer lugar, hay que especificar el panel fotovoltaico. Para ello, se utiliza la función "pvl_sapnmoduleb", la cual busca en un archivo el panel fotovoltaico seleccionado entre todos los tipos de paneles ya definidos por "Sandia National Laboratories". Esta función toma como entrada el número de la fila donde se sitúa el panel de trabajo y el archivo en el que se encuentran todos los tipos de paneles. Retorna como valores de salida las características del panel fotovoltaico seleccionado como los parámetros Isc, Voc, Imp, Vmp, etc. Seguidamente, se establecen el número de módulos en serie y en paralelo. El panel fotovoltaico definido es el mismo y cuenta con las mismas características que el diseñado en el modelo de Simulink para poder comparar los resultados que se obtienen de ambos modelos partiendo de la misma base. Este panel es el "Ligitek LM220 BB00".

En segundo lugar, se obtiene la entrada de irradiancia de un día del mes de julio del archivo CSV conseguido de la plataforma PVGIS y se define la temperatura de la celda constante a 25ºC. Así se consiguen las mismas condiciones de trabajo en los dos modelos para poder comprarlos después.

En tercer lugar, se emplea la función "pvl_sapm" para lograr los parámetros de salida. Esta función necesita como valores de entrada la estructura del módulo fotovoltaico conseguida en el primer paso, la irradiancia efectiva obtenida de la irradiancia dividida entre 1000 y la temperatura de la celda, ambas obtenidas en el segundo paso. Se generan como variables de salida la corriente de cortocircuito, la tensión de circuito abierto y la corriente, tensión y potencia en el MPP. En este ejemplo se necesitan los valores de corriente y tensión en el punto de máxima potencia, pero hay que tener cuidado, ya que estos valores son de un único módulo. Por lo tanto, para obtener la tensión y corriente resultante es necesario multiplicarlas por el número de módulos en serie y módulo en paralelo (definidos en el primer paso) respectivamente. Con los valores finales de

tensión y corriente, basta con multiplicarlos entre sí para obtener la potencia en el punto de máxima potencia.

Se asume que el módulo fotovoltaico es capaz de llegar al punto de máxima potencia y por eso no hace falta realizar ningún cálculo. En último lugar, se representan gráficamente los valores obtenidos. Se consiguen las gráficas de irradiancia, potencia, corriente y voltaje gracias a la función "plot". El código elaborado para este entorno de programación se presenta en el apartado 9.1.

7.2.1. Conexión entre Matlab y Raspberry Pi

Con el modelo de PV_LIB ya especificado, para crear código que se pueda enviar a la placa hardware es necesario convertir el archivo .m en una función con la sentencia:

```
function PVarrayModel()%#codegen
```

Y al final del archivo añadir "end" para cerrar la función. Después de añadir estas líneas al código, hay que ejecutarlo dando al botón de "Run". Posteriormente, en el "Command Window" de Matlab, se añaden las sentencias:

```
>> board = targetHardware ("Raspberry pi");
>> deploy (board, "PVL_TestScript1");
```

La función "target Hardware" crea un objeto de configuración hardware para implementar una función de Matlab en la placa y devuelve el objeto de configuración en la Raspberry Pi. Alguna de las propiedades que retorna son la dirección IP, el directorio en el que se encuentra actualmente, el usuario y la contraseña (figura 36).

```
>> board=targetHardware('Raspberry pi')

board =

targetHardware with properties:

                Name: 'Raspberry Pi'
       DeviceAddress: '192.168.0.22'
            Username: 'pi'
            Password: '*********'
            BuildDir: '/home/pi'
    EnableRunOnBoot: 0
        BuildAction: 'Build, load, and run'
        CoderConfig: [1×1 coder.CodeConfig]
```

Figura 36: Propiedades de la Raspberry Pi en Matlab

En cuanto a la función "deploy", es la encargada de implementar una función en el hardware. Tiene como parámetros de entrada, la variable donde se ha guardado el objeto de configuración hardware de la Raspberry Pi y el archivo .m con el que se trabaja. De esta forma, carga en la placa el archivo donde se ha creado el modelo del panel fotovoltaico mediante código.

La ventaja de este modelo es que se pueden modelar fácilmente los efectos de la irradiancia y temperatura a la entrada del panel fotovoltaico, en comparación con las complicaciones ofrecidas

por el modelo de Simulink en el que no se podía añadir la variable de la temperatura como entrada. No obstante, uno de los inconvenientes de este método es que no se puede añadir los componentes como convertidores, batería o algoritmos de seguimiento para lograr el modelo completo de la planta solar fotovoltaica, puesto que las funciones para modelarlos no están definidas. Otro de los inconvenientes, es que no se puede establecer la conexión con la Raspberry Pi debido a que la generación de código de Matlab no soporta las funciones de PV_LIB ni la función de ThingSpeak. En consecuencia, solo se puede visualizar el comportamiento de este modelo en Matlab y no sería una representación fiel de toda la planta solar fotovoltaica, sino que solamente lo sería del panel solar.

8. Casos de uso y resultados obtenidos

8.1. Modelo del circuito eléctrico de un panel solar fotovoltaico

El modelo (CircuitoCelda.slx) empleado en esta comparativa entre PV_LIB, Simulink y Raspberry Pi es el circuito eléctrico de un panel solar, para validar el funcionamiento del circuito desarrollado. Todos los modelos van a tener los mismos parámetros y la misma entrada de irradiancia. Aunque PV_LIB permita introducir la temperatura como entra, en el modelo diseñado en Simulink no está esa opción, por lo que solo se trabaja con la irradiancia. En la figura 37 se puede observar el diagrama de bloques que corresponde con este modelo, en el cual, se muestra la entrada (irradiancia) y las salidas (corriente, voltaje y potencia del módulo fotovoltaico).

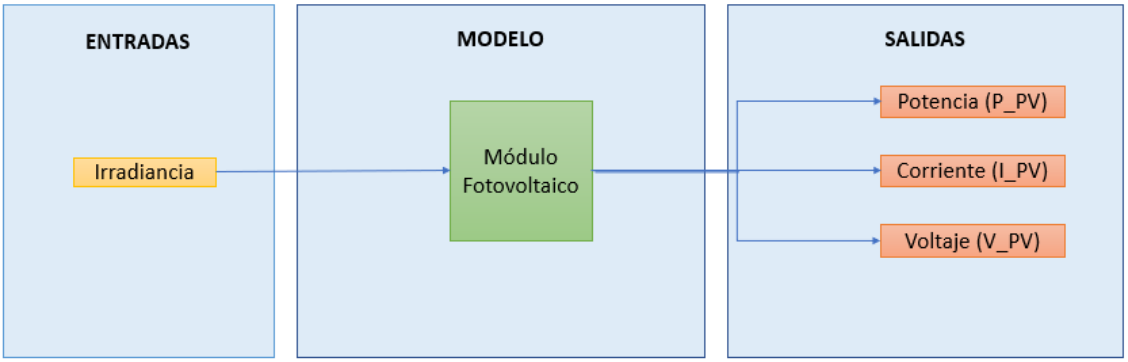

Figura 37: Diagrama de bloques del modelo CircuitoCelda

El modelo utilizado está configurado con los parámetros del "circuito eléctrico de un panel solar fotovoltaico" explicados en el apartado 6.4. Además, en Simulink, hay que añadir una carga a este circuito, la cual se modela a través de una resistencia que tiene como valor la tensión en el punto de máxima potencia entre la corriente en el punto de máxima potencia (29,2/7,54 = 3,87Ω). De esta forma se consiguen unas condiciones de carga óptimas para que el modelo alcance el MPP. Como estos valores en el punto de máxima potencia se corresponden con una irradiancia de 1000W/m^2, la entrada de los modelos será una entrada constante de 1000W/m^2 para validar con PV_LIB (ya que se corresponde con un panel en concreto y cuenta con un modelo más preciso) si el circuito eléctrico que se ha creado en Simulink se ha diseñado correctamente. Gráficamente el modelo desarrollado se puede visualizar en la figura 38. Este modelo, convertido en código C, será el que se ejecute también en la Raspberry Pi, con las sentencias mostradas en la figura 35. En cuanto al modelo desarrollado en PV_LIB, se sigue el procedimiento descrito en el apartado 7.2.

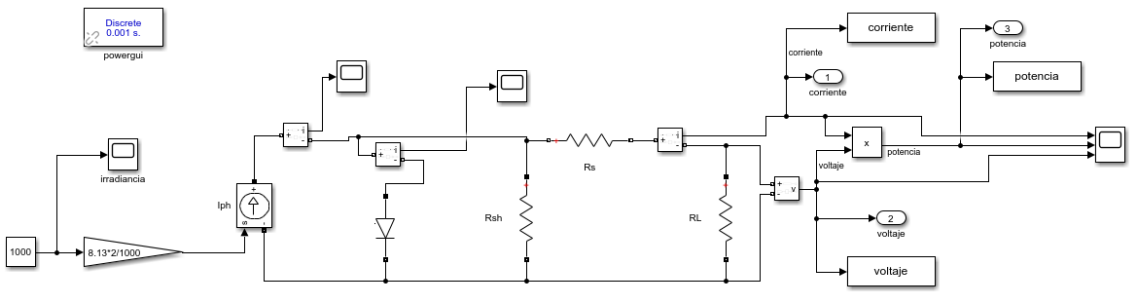

Figura 38: Modelo de un circuito eléctrico de un panel solar fotovoltaico con carga

Para una entrada constante de 1000W/m^2, el modelo ejecutado en PV_LIB, Simulink y la Raspberry Pi se obtienen unas salidas de voltaje, corriente y potencia como las mostradas en las figuras 39, 40 y 41, respectivamente.

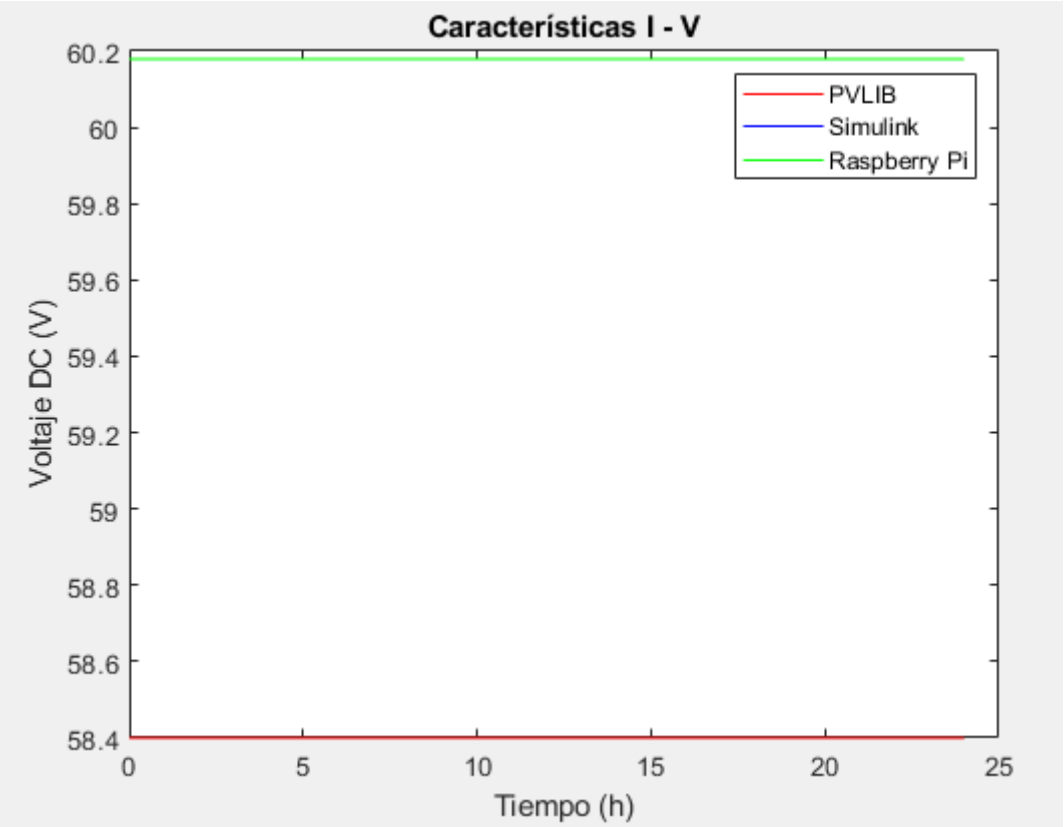

Figura 39: Comparación de Vmp entra las salidas obtenidas del modelo sencillo

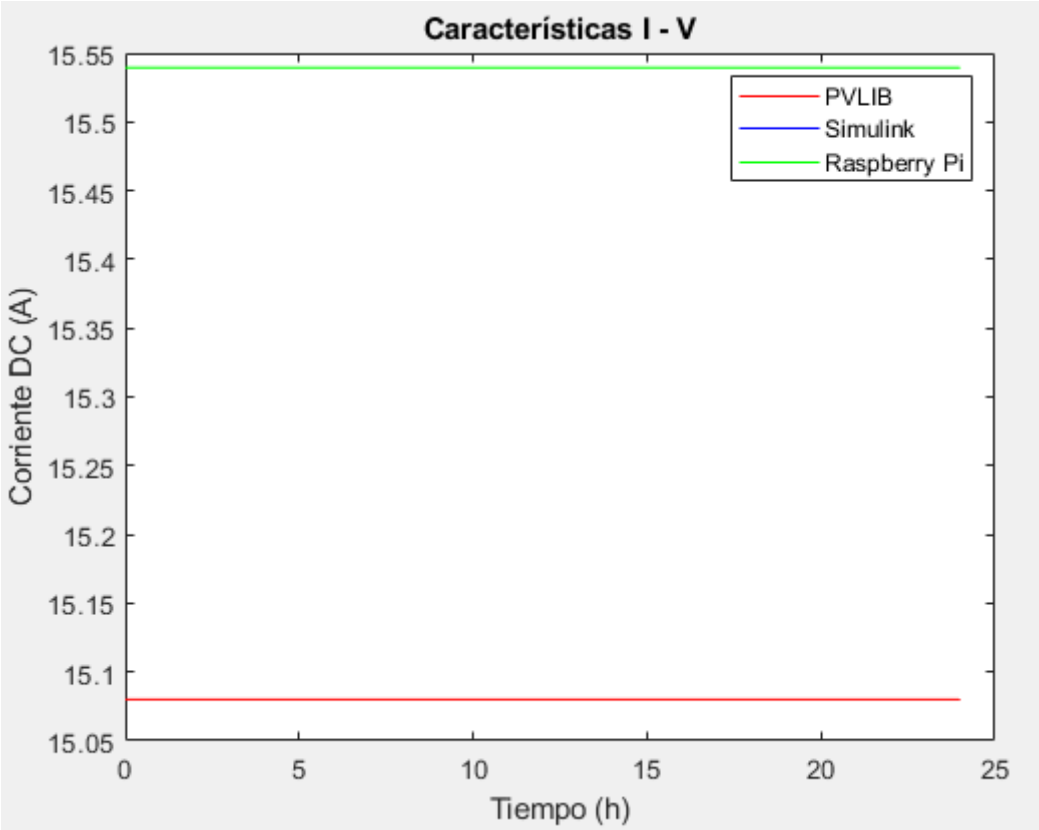

Figura 40: Comparación de Imp entre las salidas obtenidas del modelo sencillo

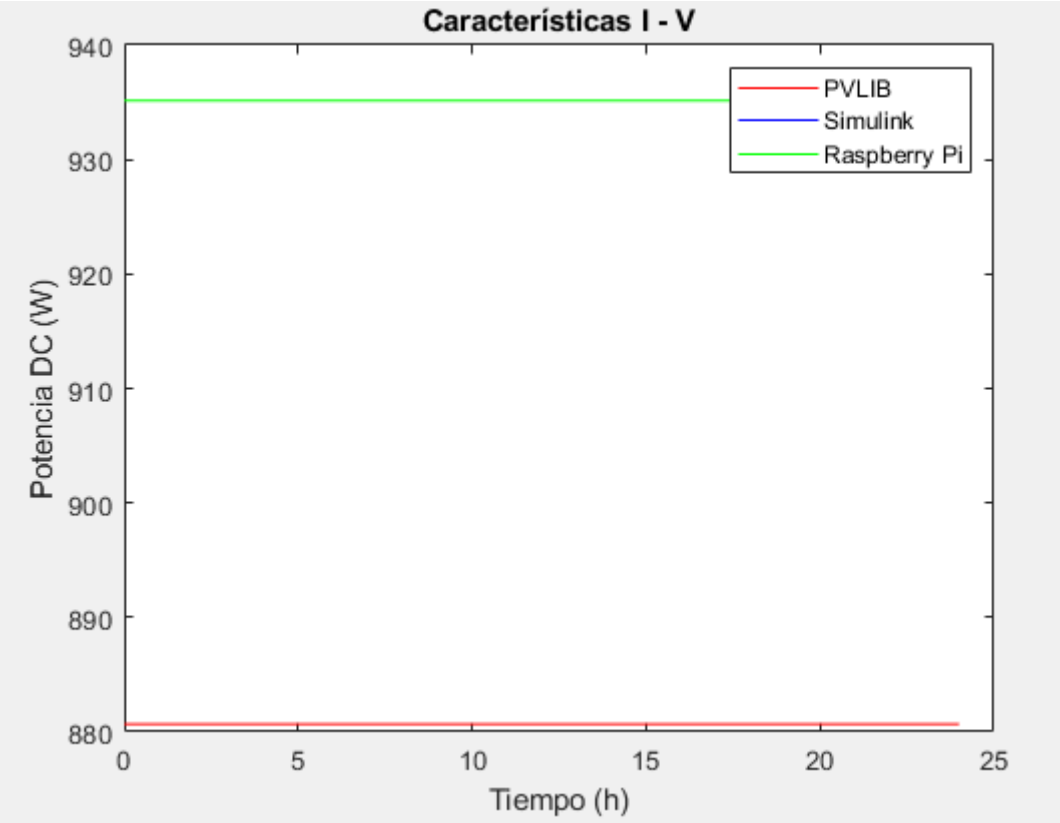

Figura 41: Comparación de Pmp entre las salidas obtenidas del modelo sencillo

Se puede comprobar que las salidas obtenidas de Simulink y de la Raspberry Pi en las tres gráficas son idénticas. Esto se debe a que el gemelo se ejecuta en tiempo real, ya que Simulink tiene un periodo de muestreo de 1ms y la Raspberry Pi tarda en ejecutar cada muestra un tiempo que oscila entre 3 – 9 µs. Como la placa hardware tiene un periodo de muestreo menor que Simulink, el gemelo digital se está ejecutando en tiempo real.

En cuanto a los datos alcanzados, se tienen que lograr unos valores en el punto de máxima potencia de Vmp = 58,40V, Imp = 15,08A, Pmp = 880,67W. Estos valores son exactamente los que se obtienen de las salidas del modelo de PV_LIB, comprobando así como el modelo desarrollado en esta Toolbox trabaja correctamente. Sin embargo, Simulink obtiene unos valores más altos (Vmp = 60,18V, Imp = 15,54A, Pmp = 935,15W), ya que a pesar de representar las pérdidas del circuito mediante la resistencia en serie y la resistencia en paralelo, hay diferencias en los parámetros con respecto a los del modelo de PV_LIB. Esto se debe a que el modelo de PV_LIB es mucho más preciso que el de Simulink, al contar con todas las variables y no linealidades del panel solar.

8.2. Modelo completo de una planta solar fotovoltaica

Para poder visualizar los efectos de una instalación solar fotovoltaica entera (con el panel fotovoltaico, el convertidor, la batería, el regulador y el MPPT), se estudia la comparativa del modelo (Circuito_completo.slx) presentado en el apartado “Modelo DT de una planta solar fotovoltaica” (figura 15). Aunque con PV_LIB no se puedan definir todos los componentes mediante las funciones que proporciona esta Toolbox, se puede emplear en la comparativa con Simulink y su implementación en la Raspberry Pi, ya que se asume que el panel fotovoltaico alcanza el MPP. Como este es un modelo más complejo que el presentado en el apartado anterior, hay que estudiar si la placa puede seguir ejecutando el gemelo digital en tiempo real. Se resumen

las entradas, el modelo y las salidas con las que se trabaja en el diagrama de bloques de la figura 42.

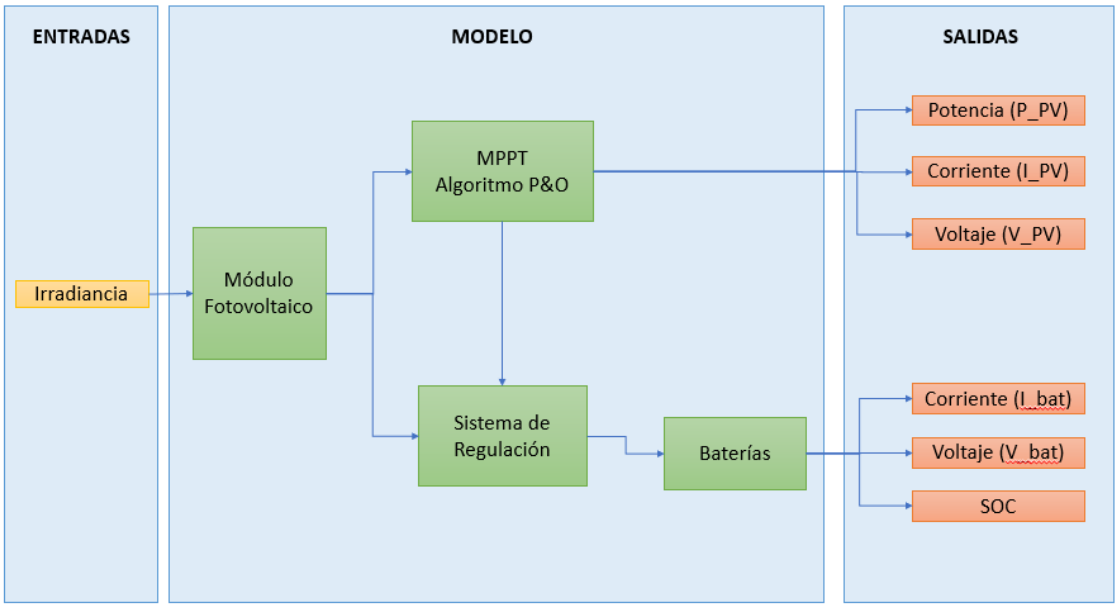

Figura 42: Diagrama de bloques del modelo Circuito_completo

En este apartado se trabaja con la irradiancia obtenida del archivo de PVGIS con los datos promedios de irradiancia para un día del mes de julio. Se consigue una entrada de 24 muestras, una por cada hora del día como se puede ver en la figura 43.

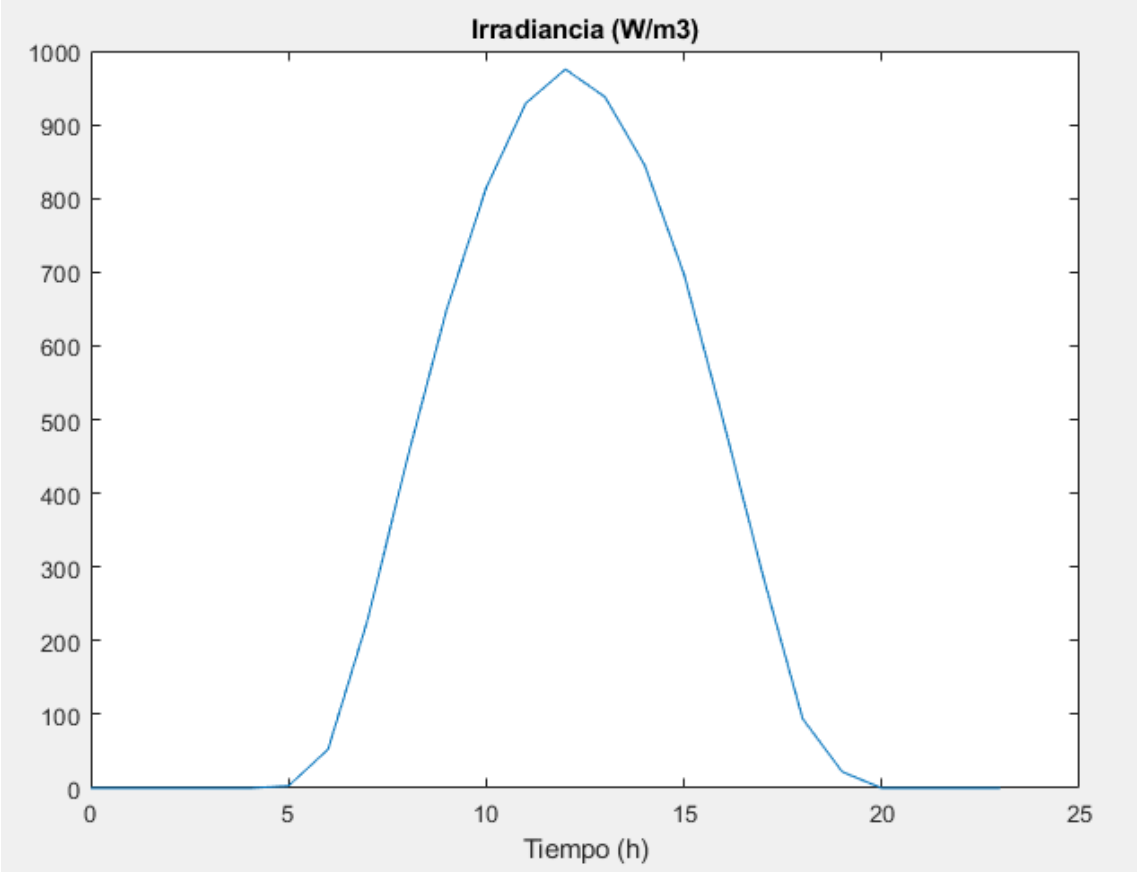

Figura 43: Parámetro de entrada: irradiancia

Para la entrada de la figura 43, se obtienen las salidas de voltaje (figura 44), corriente (figura 45) y potencia (figura 46) del panel mostradas a continuación:

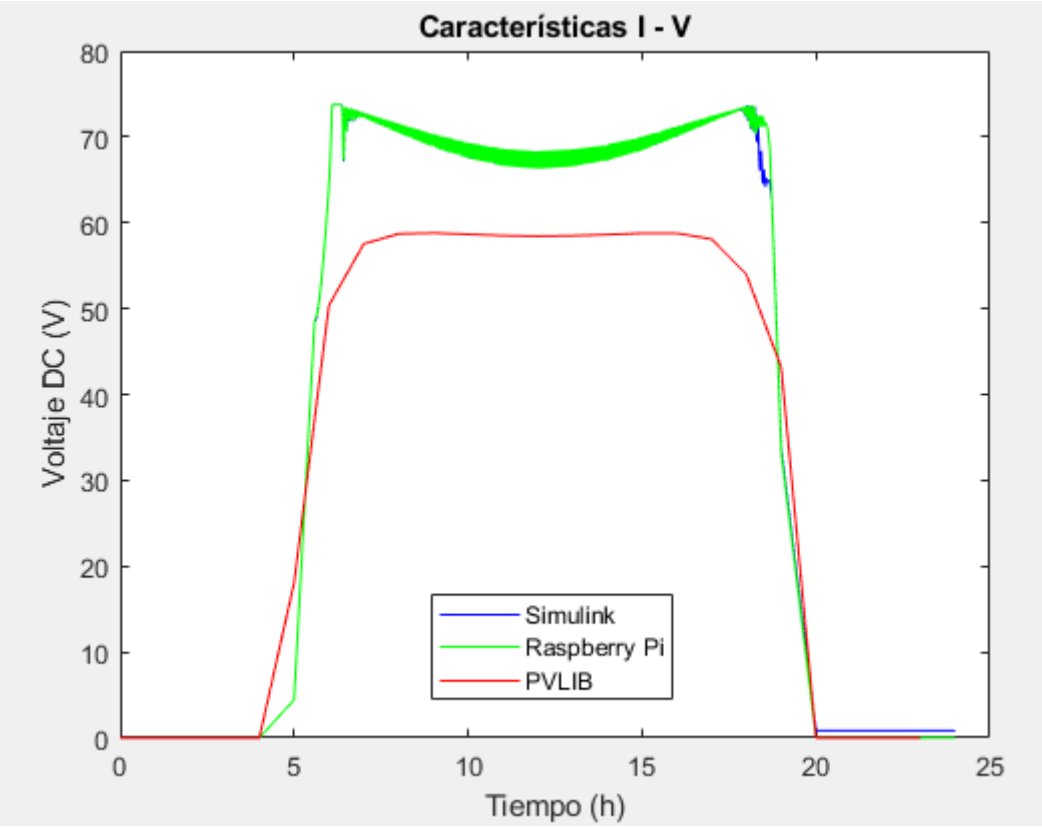

Figura 44: Comparación de Vmp entre las salidas del modelo complejo

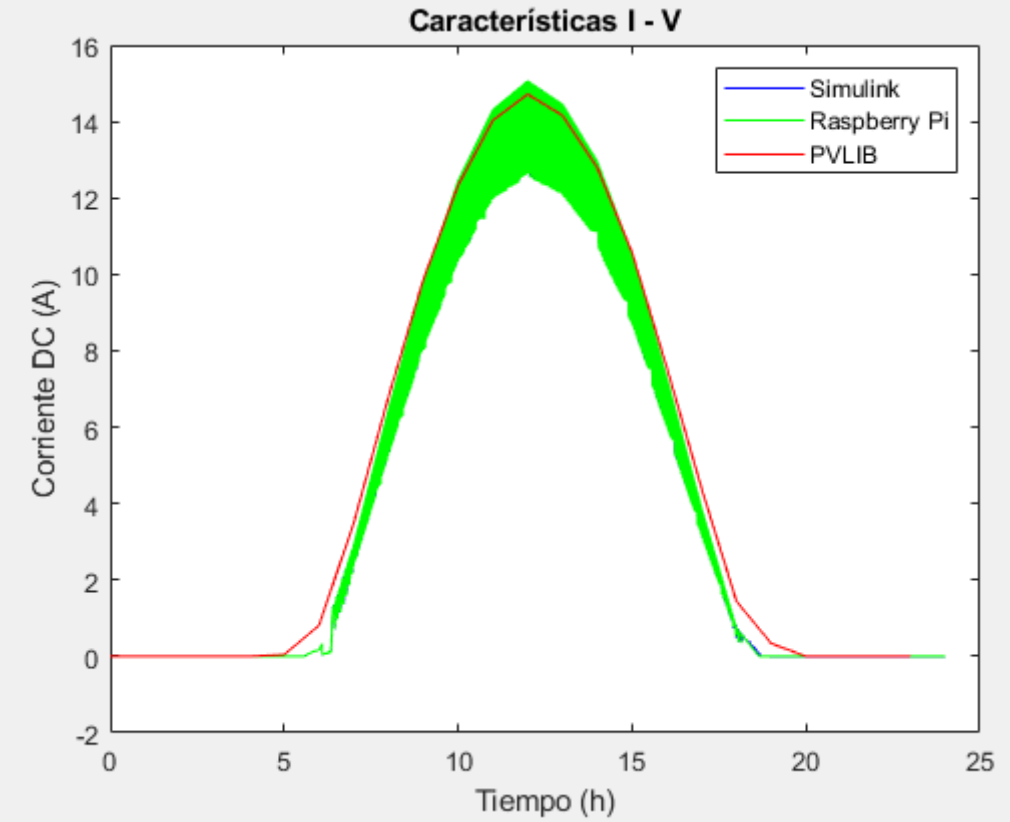

Figura 45: Comparación de Imp entre las salidas del modelo complejo

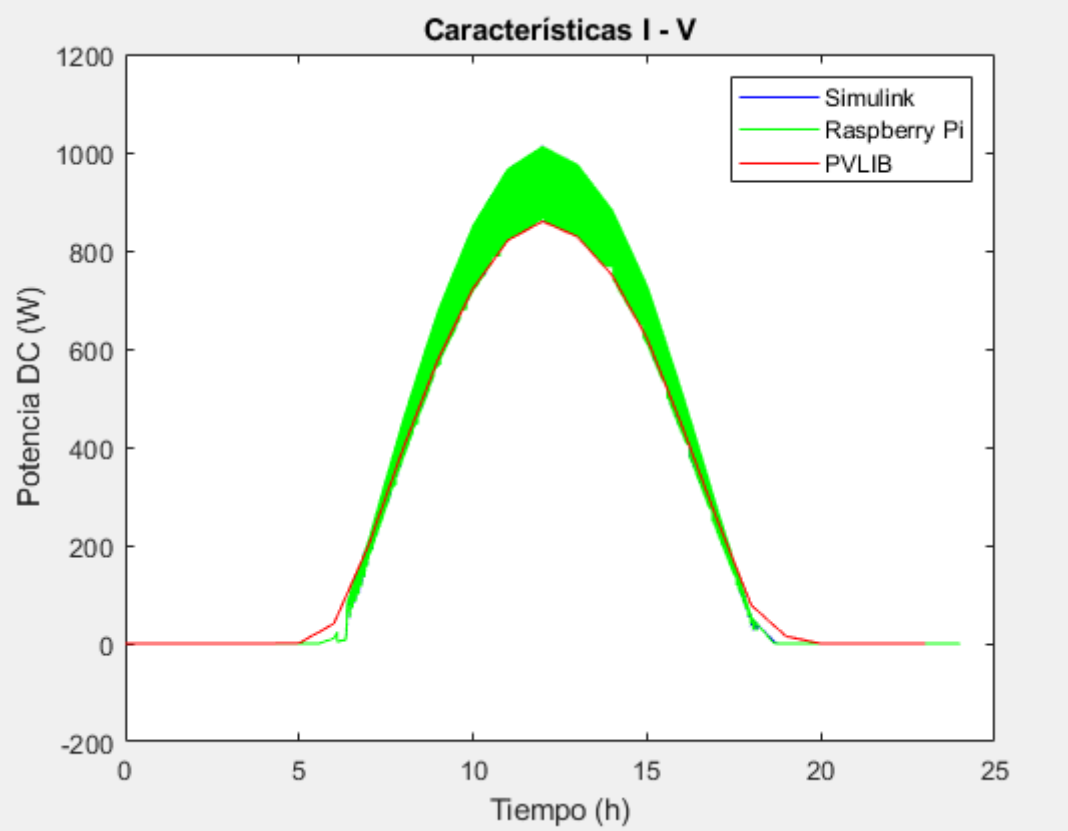

Figura 46: Comparación de Pmp entre las salidas del modelo complejo

Como se puede ver en las gráficas, las salidas que ofrece Simulink y la Raspberry Pi son prácticamente iguales. El modelo de Simulink tiene un periodo de muestreo de 10 µs (establecido por defecto) y el modelo de la Raspberry Pi tarda entre 6 -10 µs en ejecutar cada muestra del modelo. Aunque el gemelo digital se sigue ejecutando en tiempo real, hay muestras que no coinciden con las de Simulink porque el tiempo de ejecución de la Raspberry Pi es mayor que el periodo de muestreo. Esto es debido a la complejidad del modelo, lo que significa que el coste de procesamiento es mayor y la Raspberry Pi tarda más en llevarlo a cabo.

También, se puede contemplar que en las tres gráficas hay cierta dispersión en los resultados de Simulink y Raspberry Pi, debido a que con el algoritmo de perturbación y observación genera oscilaciones alrededor del punto de máxima potencia. Asimismo, se nota una diferencia entre los valores obtenidos de Simulink y Raspberry Pi con los valores obtenidos de PV_LIB. Esto se debe a la diferencia entre los parámetros de ambos modelos, al igual que las diferencias obtenidas en los resultados del circuito eléctrico de un panel solar.

La gráfica de la corriente (figura 45) tiene la misma tendencia que la curva de la irradiancia (figura 43), puesto que depende directamente de ella. Sin embargo la curva del voltaje (figura 44), frente a grandes variaciones de irradiancia se generan pequeños incrementos de voltaje, por lo que se puede apreciar una gráfica más o menos constante durante las horas de más sol. Por último, la gráfica de la potencia (figura 46) se alcanza como la multiplicación de la corriente y la tensión, formando una curva que tiene la misma tendencia que la irradiancia.

También se muestran las salidas de SOC (figura 47) corriente (figura 48) y voltaje (figura 49) de la batería.

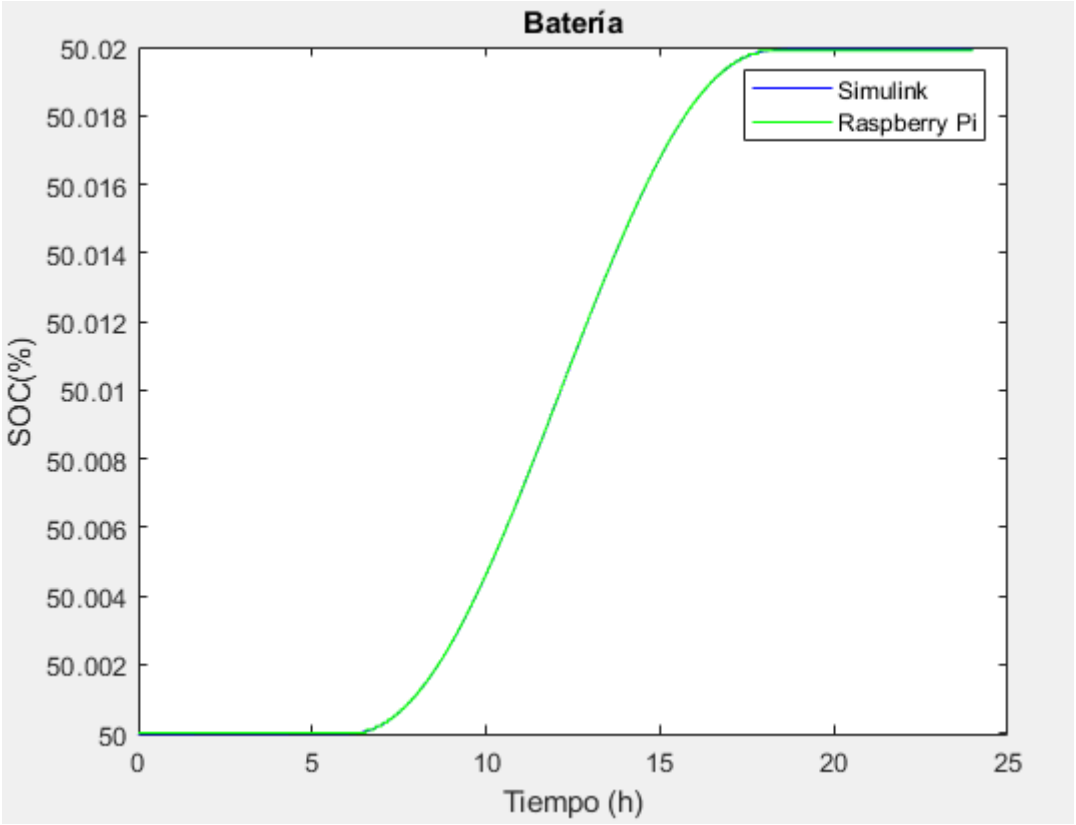

Figura 47: Comparación de SOC en la batería entre Simulink y Raspberry Pi

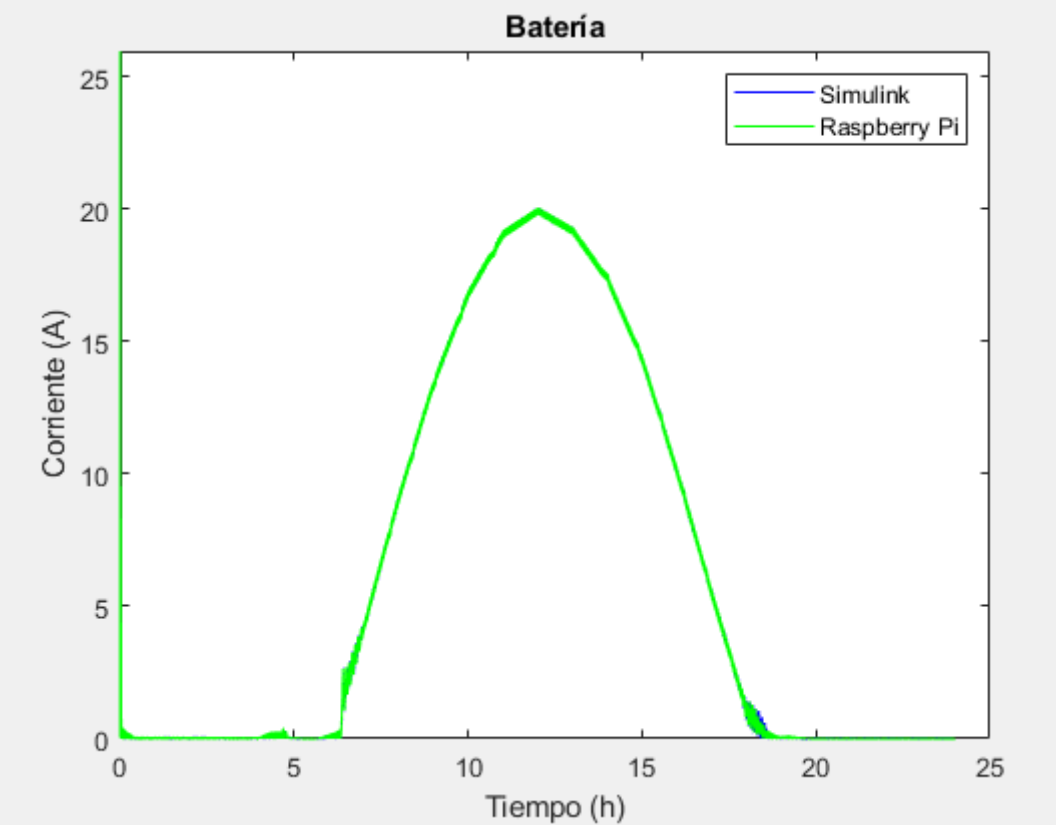

Figura 48: Comparación de corriente en la batería entre Simulink y Raspberry Pi

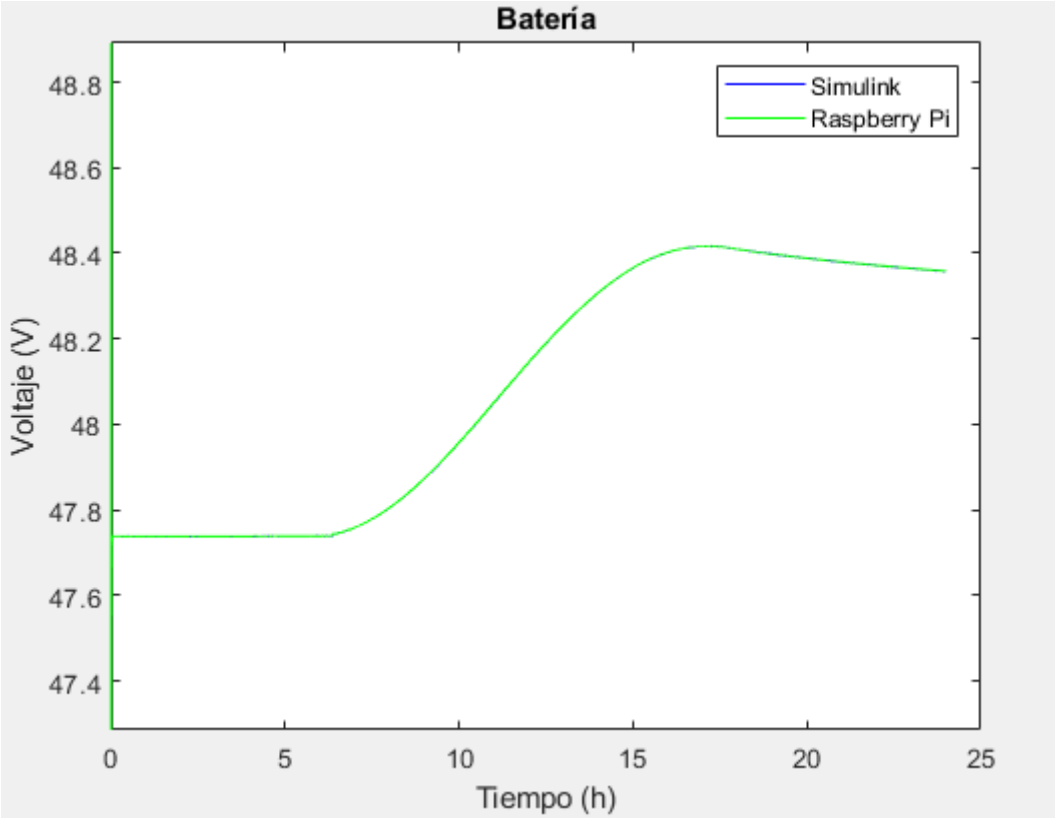

Figura 49: Comparación de voltaje en la batería entre Simulink y Raspberry Pi

La batería parte del estado inicial de carga igual al 50%, es decir, inicialmente la batería esta carga a la mitad y se carga hasta el 50,02% (figura 47). En la figura 48 se muestra las horas en la que la batería está almacenando la corriente para poder utilizarla en las horas que no haya luz. Por último, la gráfica del voltaje (figura 49) sigue la misma tendencia que la curva de carga de la batería, se puede observar cómo aumenta el voltaje cuando se está cargando la batería y se mantiene constante cuando el estado de carga se mantiene constante.

9. Conclusiones y trabajo futuro

De las salidas obtenidas con ambos modelos se presenta la tabla 1 para comentar las diferencias entre el modelo simple y el modelo complejo.

Comparativa	Modelo simple	Modelo complejo
Tiempo de ejecución	Simulink cuenta con un periodo de muestreo de 1ms y la Raspberry Pi tarda en ejecutar el modelo entre 3 – 9µs. El gemelo digital se ejecuta en tiempo real.	Simulink cuenta con un periodo de muestreo de 10 µs y la Raspberry Pi tarda en ejecutar el modelo entre 6 - 10µs. El gemelo digital se ejecuta en tiempo real, salvo en alguna muestra debido a que el periodo de muestreo de Simulink es muy pequeño.
Métodos	Las señales de salida obtenidas de Simulink y la Raspberry Pi son idénticas.	Las salidas obtenidas de Simulink y Raspberry Pi son prácticamente iguales aunque difieren en alguna muestra en el intervalo de 18h a 19h.
Corriente	PV_LIB Imp = 15,08A Simulink y Raspberry Pi Imp = 15,54A	En todo los casos, a mayor irradiancia, mayor corriente. Además, la curva de la corriente sigue la misma tendencia que la irradiancia ya que depende directamente de ella.
Tensión	PV_LIB Vmp = 58,40V Simulink y Raspberry Pi Vmp = 60,18V	En todos los casos, a mayor irradiancia mayor tensión, pero se generan unos incrementos de tensión menores que los de la corriente. Por eso, la gráfica se mantiene más o menos constante en las horas de sol.
Potencia	PV_LIB Pmp = 880,67W Simulink y Raspberry Pi Pmp = 935,15W	En todos los casos, se consigue de la multiplicación de la corriente y la tensión.

Tabla 1: Comparación final de los modelos

En este TFG se ha creado el gemelo digital de la planta solar fotovoltaica que se encuentra en el Edificio Politécnico de la Universidad de Alcalá de Henares. El desarrollo del DT se ha elaborado en Simulink y con la Toolbox PV_LIB, para posteriormente ejecutarlo en la Raspberry Pi 4 model B. Una vez implementado el modelo, se ha hecho una comparación entre los diferentes entornos para ver las ventajas y los inconvenientes de los mismos.

En la generación del gemelo digital se ha comprobado su difícil implementación, puesto que, al ser una tecnología nueva todavía cuenta con muchas limitaciones. Se han ido encontrado numerosas problemáticas a la hora de implementar el gemelo, como los problemas surgidos para encontrar un buen modelo (debido al problema del bloque PV array), la obtención de código, la comunicación con ThingSpeak y la ejecución en tiempo real. Por una parte, el mayor problema de todos ha sido la conversión del modelo a código C. En primer lugar, en Simulink no se podían

usar todos los bloques debido a que algunos imposibilitaban que se pudieran compilar los archivos generados. En segundo lugar, en PV_LIB no se ha conseguido generar código C porque las funciones empleadas no eran compatibles con la generación del código. Además, en ambos casos con el bloque de ThingSpeak para la Raspberry Pi en Simulink y la función de ThingSpeak en PV_LIB, no se ha podido generar código C. Por otra parte, los inconvenientes para encontrar un buen modelo han sido ocasionados principalmente por el bloque PV array, ya que en las actuales versiones de Matlab no son capaces de simular el modelo. Por último, se he demostrado que cuanto más complejo sea un modelo mayor es su coste de procesamiento. Por lo tanto, si el modelo cuenta con un tiempo de muestreo pequeño y es complejo, seguramente no se pueda ejecutar el gemelo digital en tiempo real debido a su alto coste de procesamiento.

En definitiva, el gemelo digital en sistemas de generación de energía solar fotovoltaica presenta numerosos beneficios como el análisis de datos y detección de fallos, pero aún sigue en su fase de desarrollo, por lo que todavía cuenta con algunos problemas de implementación. Por ello, se debe contar con un buen modelo y un equipo potente que sea capaz de procesar todos los datos que recibe en tiempo real.

Tras elaborar este proyecto se identifican nuevas líneas de trabajo.

Una vez obtenidas las salidas de la planta, como potencia, tensión y corriente del panel y corriente y tensión en la batería, se pueden comparar los resultados obtenidos del modelo virtual con los que consigue la planta real. De esta forma se podría ver las diferencias que existan entre ambas y ajustar el modelo virtual para que se asemeje lo máximo posible al modelo físico.

Para desarrollar el modelo, en vez de utilizar Matlab, existen otras opciones como Modelica y PV_LIB para Python. La primera opción es un lenguaje de modelado, orientada a modelar elementos de sistemas complejos, que tiene la ventaja de poder compilarse directamente a diferencia de Simulink, donde primero hay que crear el código y modificarlo para posteriormente compilarlo en la Raspberry Pi. La segunda opción sería una buena alternativa de PV_LIB para Matlab, puesto que con la de Matlab no se consigue generar código que se pueda implementar en la placa hardware. Aun así, habría que comprobar que este entorno cumple con los requisitos de tiempo real para el desarrollo del gemelo digital. Aunque PV_LIB para Matlab y para Python difieren en estructura y contenido, se basan en funciones que modelan la potencia de los sistemas de generación de energía solar fotovoltaica.

10. Planos y diagramas

10.1. Código del modelo CircuitoCelda

Se exponen los códigos de Simulink y de Matlab para este modelo.

1. Simulink

Al crear código, Simulink genera una carpeta (CircuitoCelda_ert_rtw) que contiene varios archivos, de la cual solo se trabaja con el main.c. Inicialmente, este archivo parte de la inicialización del modelo con la función "CircuitoCelda_initialize()" y de la ejecución del mismo con la función "rt_OneStep()". Para poder guardas las salidas en un archivo y ver el tiempo de ejecución del modelo se han tenido que realizar algunos cambios, como modificar el main para mostrar las salidas y añadir las librerías necesarias para su correcto funcionamiento.

Se añaden las librerías math.h para que pueda ejecutarse, "time.h" para que se pueda obtener el tiempo de ejecución y "stdlib.h" para poder guardas las variables en un archivo de texto.

Cambios realizados en el main por orden:

- Se definen unas variables auxiliares: "a" para llevar la cuenta de las muestras que genera el programa, "t0" y "t1" se utilizar para guardar el valor del instante de tiempo en el que se ejecutan y "tiempo" para obtener el tiempo final de ejecución.
- Se crea un fichero de texto "aux.txt" en el que se guardan las variables de salida.
- Para que la aplicación se esté ejecutando repetidamente se utiliza un while que como máximo se repite el mismo número de muestra que genera el archivo de Simulink para poder compararlos después. Si se quiere obtener una aplicación que se repita indefinidamente se tendría que ejecutar dentro de un while(1). Dentro del bucle se ejecuta la función rt_OneStep y el tiempo que tarda en ejecutarse. Para ello, con la función "clock" se obtiene el tiempo antes y después de que se ejecute, se resta y se pasa de tics a segundos con la función "CLOCKS_PER_SEC". Asimismo, se escribe en el archivo de texto las salidas de corriente, voltaje y potencia del módulo fotovoltaico y el tiempo de ejecución.
- Finalmente se cierra el archivo.

```
/*
 * Academic License - for use in teaching, academic research, and meeting
 * course requirements at degree granting institutions only.  Not for
 * government, commercial, or other organizational use.
 *
 * File: ert_main.c
 *
 * Code generated for Simulink model "CircuitoCelda".
 *
 * Model version                  : 1.13
 * Simulink Coder version         : 9.7 (R2022a) 13-Nov-2021
 * C/C++ source code generated on : Sun Sep 18 19:19:13 2022
 *
 * Target selection: ert.tlc
 * Embedded hardware selection: ARM Compatible->ARM Cortex-A
 * Code generation objectives:
 *    1. Execution efficiency
```

```
 *    2. RAM efficiency
 * Validation result: Not run
 */

#include <stddef.h>
#include <stdio.h>           /* This example main program uses printf/fflush */
#include "CircuitoCelda.h"           /* Model header file */
#include <math.h>
#include <time.h>
#include <stdlib.h>

/*
 * Associating rt_OneStep with a real-time clock or interrupt service routine
 * is what makes the generated code "real-time".  The function rt_OneStep is
 * always associated with the base rate of the model.  Subrates are managed
 * by the base rate from inside the generated code.  Enabling/disabling
 * interrupts and floating point context switches are target specific.  This
 * example code indicates where these should take place relative to executing
 * the generated code step function.  Overrun behavior should be tailored to
 * your application needs.  This example simply sets an error status in the
 * real-time model and returns from rt_OneStep.
 */
void rt_OneStep(void);
void rt_OneStep(void)
{
  static boolean_T OverrunFlag = false;

  /* Disable interrupts here */

  /* Check for overrun */
  if (OverrunFlag) {
    rtmSetErrorStatus(rtM, "Overrun");
    return;
  }

  OverrunFlag = true;

  /* Save FPU context here (if necessary) */
  /* Re-enable timer or interrupt here */
  /* Set model inputs here */

  /* Step the model */
  CircuitoCelda_step();

  /* Get model outputs here */

  /* Indicate task complete */
  OverrunFlag = false;

  /* Disable interrupts here */
  /* Restore FPU context here (if necessary) */
  /* Enable interrupts here */
}

/*
```

```
 * The example main function illustrates what is required by your
 * application code to initialize, execute, and terminate the generated code.
 * Attaching rt_OneStep to a real-time clock is target specific. This example
 * illustrates how you do this relative to initializing the model.
 */
int_T main(int_T argc, const char *argv[])
{
  /* Unused arguments */
  (void)(argc);
  (void)(argv);

  /* Initialize model */
  CircuitoCelda_initialize();

  /* Attach rt_OneStep to a timer or interrupt service routine with
   * period 0.001 seconds (base rate of the model) here.
   * The call syntax for rt_OneStep is
   *
   *  rt_OneStep();
   */
  double t0, t1; //variables auxiliares para calcular el tiempo de ejecución
  double tiempo; //variable donde se guarda el tiempo de ejecución
  int a = 0;     //variable que lleva la cuenta de las muestras

  FILE* fichero;
  fichero = fopen("aux.txt", "a+");
  fprintf(fichero, "muestra\t i_pv\t v_pv\t p_pv\t tiempo\n");

  while (a <= 24000) {

     a = a + 1;
     t0 = clock();
     rt_OneStep();
     t1 = clock();
     tiempo = (t1 - t0) / CLOCKS_PER_SEC;         //pasamos de tics a segundos

     fprintf(fichero, "%d %.4f\t", a, rtY.corriente);
     fprintf(fichero, "%.4f\t", rtY.voltaje);
     fprintf(fichero, "%.4f\t", rtY.potencia);
     fprintf(fichero, "%lf s\n", tiempo);

  }
  fclose(fichero);
  return 0;
}

/*
 * File trailer for generated code.
 *
 * [EOF]
 */
```

2. Comparación de los modelos

Para comparar las salidas obtenidas de Simulink, PV_LIB y la Raspberry Pi se crea el script "comparacion_CircuitoCelda.m". En este archivo se establecen el desarrollo del modelo en PV_LIB con sus entradas correspondientes, la lectura de los datos obtenidos de la Raspberry Pi, la ejecución del archivo "CircuitoCelda.slx" en Simulink y la comparativa final entre todos mediante gráficas.

Partes en el desarrollo de este script:

- Ejecución de los modelos

En primer lugar, se desarrolla el modelo de PV_LIB (sus funciones y funcionamiento ya se han comentado en el apartado 7.2). A modo resumen, se elige el panel "Ligitek Photovoltaic LM220BB00" de la base de "Sandia National Laboratories" y se definen sus características, se establecen los parámetros de entrada (la irradiancia constante a 1000W/m^2 y la temperatura de la celda constante a 25ºC) y finalmente se obtienen los valores de corriente, tensión y potencia total del panel. En segundo lugar, se leen los datos obtenidos de la Raspberry Pi, guardados en el archivo de texto "copia.txt". En tercer lugar, se ejecuta el archivo de Simulink con la función "sim".

- Comparación de las salidas

Por una parte, se crean las gráficas de la irradiancia en función del tiempo, para poder ver visualmente la entrada al panel. Por otra parte, se generan las gráficas de las salidas del panel, corriente, voltaje y potencia en función del tiempo para los tres métodos empleados.

```
%EJECUCIÓN DE LOS MODELOS

% Crear modelo PV_LIB
ModuleParameters = pvl_sapmmoduledb(181,"SandiaModuleDatabase_20120925.xlsx"); % Se define the PV module
Array.Ms = 2; % Número de módulos en serie
Array.Mp = 2; % Número de módulos en paralelo

%time = (0:1:23)";
irradiancia = linspace(1000,1000,24001)";
Ee = irradiancia/1000;          %irradiancia efectiva
Tcel = linspace(25,25,24001)";

mSAPMResults = pvl_sapm(ModuleParameters, Ee, Tcel);
aSAPMResults.Vmp = Array.Ms * mSAPMResults.Vmp;
aSAPMResults.Imp = Array.Mp * mSAPMResults.Imp;
aSAPMResults.Pmp = aSAPMResults.Vmp .* aSAPMResults.Imp;

%Leer los datos de la Raspberry del archivo de texto
valoresPi = readtable ("copia.txt");
i_placa = valoresPi.Var2;
v_placa = valoresPi.Var3;
p_placa = valoresPi.Var4;

%Se ejecuta el archivo de Simulink
sim("CircuitoCelda.slx");

%COMPARACIÓN DE LAS SALIDAS
```

```
% Se muestra las caracterísitcas I - V de un panel solar fotovoltaico
% Comparativa entre Simulink, PV_LIB y Raspberry Pi

figure
plot(tout,aSAPMResults.Pmp,”-r”)
hold on
plot(tout,potencia,”-b”)
hold on
plot(tout,p_placa,”-g”)
legend({“PVLIB”,”Simulink”,”Raspberry Pi”})
hold off
ylabel(“Potencia DC (W)”)
xlabel(“Tiempo (h)”)
title(“Características I - V”)

figure
plot(tout,aSAPMResults.Imp,”-r”)
hold on
plot(tout,corriente,”-b”)
hold on
plot(tout,i_placa,”-g”)
legend({“PVLIB”,”Simulink”,”Raspberry Pi”})
hold off
ylabel(“Corriente DC (A)”)
xlabel(“Tiempo (h)”)
title(“Características I - V”)

figure
plot(tout,aSAPMResults.Vmp,”-r”)
hold on
plot(tout,voltaje,”-b”)
hold on
plot(tout,v_placa,”-g”)
legend({“PVLIB”,”Simulink”,”Raspberry Pi”})
hold off
ylabel(“Voltaje DC (V)”)
xlabel(“Tiempo (h)”)
title(“Características I - V”)
```

10.2. Código del modelo Circuito_completo

Al igual que el apartado anterior se exponen los archivos obtenidos en Simulink y Matlab.

1. Simulink

Al crear el código C, Simulink genera la carpeta Circuito_completo_ert_rtw de la que se trabaja con el archivo main.c. El funcionamiento y desarrollo del código es igual que el del modelo más sencillo. La única diferencia que tiene este código es que ahora se guardan también las salidas de corriente, voltaje y SOC de la batería.

```
/*
 * Academic License - for use in teaching, academic research, and meeting
 * course requirements at degree granting institutions only.  Not for
```

```
 * government, commercial, or other organizational use.
 *
 * File: ert_main.c
 *
 * Code generated for Simulink model “Circuito_completo”.
 *
 * Model version                  : 6.6
 * Simulink Coder version         : 9.7 (R2022a) 13-Nov-2021
 * C/C++ source code generated on : Mon Sep 19 15:46:37 2022
 *
 * Target selection: ert.tlc
 * Embedded hardware selection: ARM Compatible->ARM Cortex-A
 * Code generation objectives:
 *    1. Execution efficiency
 *    2. RAM efficiency
 * Validation result: Not run
 */

#include <stddef.h>
#include <stdio.h>            /* This example main program uses printf/fflush */
#include "Circuito_completo.h"         /* Model header file */
#include <math.h>
#include <time.h>
#include <stdlib.h>

/*
 * Associating rt_OneStep with a real-time clock or interrupt service routine
 * is what makes the generated code "real-time".  The function rt_OneStep is
 * always associated with the base rate of the model.  Subrates are managed
 * by the base rate from inside the generated code.  Enabling/disabling
 * interrupts and floating point context switches are target specific.  This
 * example code indicates where these should take place relative to executing
 * the generated code step function.  Overrun behavior should be tailored to
 * your application needs.  This example simply sets an error status in the
 * real-time model and returns from rt_OneStep.
 */
void rt_OneStep(void);
void rt_OneStep(void)
{
  static boolean_T OverrunFlag = false;

  /* Disable interrupts here */

  /* Check for overrun */
  if (OverrunFlag) {
    rtmSetErrorStatus(rtM, "Overrun");
    return;
  }

  OverrunFlag = true;

  /* Save FPU context here (if necessary) */
  /* Re-enable timer or interrupt here */
  /* Set model inputs here */
```

```
  /* Step the model */
  Circuito_completo_step();

  /* Get model outputs here */

  /* Indicate task complete */
  OverrunFlag = false;

  /* Disable interrupts here */
  /* Restore FPU context here (if necessary) */
  /* Enable interrupts here */
}

/*
 * The example main function illustrates what is required by your
 * application code to initialize, execute, and terminate the generated code.
 * Attaching rt_OneStep to a real-time clock is target specific. This example
 * illustrates how you do this relative to initializing the model.
 */
int_T main(int_T argc, const char *argv[])
{
  /* Unused arguments */
  (void)(argc);
  (void)(argv);

  /* Initialize model */
  Circuito_completo_initialize();

  /* Simulating the model step behavior (in non real-time) to
   *  simulate model behavior at stop time.
   */
  double t0, t1; //variables auxiliares para calcular el tiempo de ejecución
  double tiempo; //variable donde se guarda el tiempo de ejecución
  int a = 0;     //variable que lleva la cuenta de las muestras

  FILE* fichero;
  fichero = fopen("aux.txt", "a+");
  fprintf(fichero, "muestra\t i_pv\t v_pv\t p_pv\t SOCbat\t ibat\t vbat\t tiempo\n");

  while (a <= 2400000) {

     a = a + 1;
     t0 = clock();
     rt_OneStep();
     t1 = clock();
     tiempo = (t1 - t0) / CLOCKS_PER_SEC;          //pasamos de tics a segundos

     fprintf(fichero, "%d\t", a);
     fprintf(fichero, "%.4f\t", rtY.corriente);
     fprintf(fichero, "%.4f\t", rtY.voltaje);
     fprintf(fichero, "%.4f\t", rtY.potencia);
     fprintf(fichero, "%.4f\t", rtY.SOCbat);
     fprintf(fichero, "%.4f\t", rtY.ibat);
     fprintf(fichero, "%.4f\t", rtY.vbat);
     fprintf(fichero, "%lf s\n", tiempo);
```

```
  }
  fclose(fichero);
  return 0;
}

/*
 * File trailer for generated code.
 *
 * [EOF]
 */
```

2. Comparación de los modelos

El desarrollo de este código es igual al obtenido del modelo simple, con las diferencias de que antes contaba con dos partes, una para los modelos en ejecución y otra para la comparación de las salidas y ahora cuenta con tres partes, añadiendo el apartado de definición de la entrada.

- Valores de entrada

Los valores de entrada se obtienen como ya se expuso en el apartado 6.5.6, en un archivo CSV de un día del mes de Julio gracias a la plataforma de PVGIS. De los datos que vienen en el archivo CSV se trabaja solo con la irradiancia total, de la cual los últimos datos que se guardan en Matlab hay que eliminarlos ya que no pertenecen a este parámetro. También es necesario definir el vector de tiempo con el que se va a trabajar. Se establece un tiempo de 24h (para un día entero). Por último, en la variable "entrada", se recogen los valores de irradiancia y tiempo para mandarlos como entrada al modelo de Simulink.

```
%VALORES DE ENTRADA
time = (0:1:23)";
data = readtable ("irradjulio.csv");
irradiancia = data.G_i_;
% Se eliminan los últimos valores porque no pertenecen a la irradiancia
irradiancia (28,:) = []; irradiancia (27,:) = []; irradiancia (26,:) = []; irradiancia (25,:) = [];
entrada = [time irradiancia];

%EJECUCIÓN DE LOS MODELOS

% Crear modelo PV_LIB
ModuleParameters = pvl_sapmmoduledb(181,"SandiaModuleDatabase_20120925.xlsx"); % Se
define the PV module
Array.Ms = 2; % Número de módulos en serie
Array.Mp = 2; % Número de módulos en paralelo

Ee = irradiancia/1000;          %irradiancia efectiva
Tcel = linspace(25,25,24)";

mSAPMResults = pvl_sapm(ModuleParameters, Ee, Tcel);
aSAPMResults.Vmp = Array.Ms * mSAPMResults.Vmp;
aSAPMResults.Imp = Array.Mp * mSAPMResults.Imp;
aSAPMResults.Pmp = aSAPMResults.Vmp .* aSAPMResults.Imp;

%Leer los datos de la Raspberry del archivo de texto
```

```
valoresPi = readtable (“copia1.txt”);
i_placa = valoresPi.Var2;
v_placa = valoresPi.Var3;
p_placa = valoresPi.Var4;

%Se ejecuta el archivo de Simulink
sim(“Circuito_completo.slx”);

%COMPARACIÓN DE LAS SALIDAS

% Se muestra la irradiancia en función del tiempo
figure
plot(time,irradiancia)
xlabel(“Tiempo (h)”)
title(“Irradiancia (W/m3)”)

% Se muestra las caracterísitcas I - V de un panel solar fotovoltaico
% Comparativa entre Simulink y Raspberry Pi

figure
plot(tout,potencia,”-b”)
hold on
plot(tout,p_placa,”-g”)
hold on
plot(time,aSAPMResults.Pmp,”-r”)
legend({“Simulink”,”Raspberry Pi”,”PVLIB”})
hold off
ylabel(“Potencia DC (W)”)
xlabel(“Tiempo (h)”)
title(“Características I - V”)

figure
plot(tout,corriente,”-b”)
hold on
plot(tout,i_placa,”-g”)
hold on
plot(time,aSAPMResults.Imp,”-r”)
legend({“Simulink”,”Raspberry Pi”,”PVLIB”})
hold off
ylabel(“Corriente DC (A)”)
xlabel(“Tiempo (h)”)
title(“Características I - V”)

figure
plot(tout,voltaje,”-b”)
hold on
plot(tout,v_placa,”-g”)
hold on
plot(time,aSAPMResults.Vmp,”-r”)
legend({“Simulink”,”Raspberry Pi”,”PVLIB”})
hold off
ylabel(“Voltaje DC (V)”)
xlabel(“Tiempo (h)”)
title(“Características I - V”)
```

10.3. Código en PV_LIB con la entrada obtenida de ThingSpeak

El código que se muestra en este apartado es el mismo que el explicado en el apartado 7.2, pero las entradas del panel solar se obtienen de ThingSpeak, a diferencia del apartado 7.2 donde la entrada de irradiancia se obtiene de PVGIS y la temperatura de la celda es constante. Para conseguir las entradas, se crea un bucle en el que se va a estar leyendo los valores de irradiancia y temperatura de ThingSpeak y se van guardando en un array para posteriormente poder visualizarlos. En este paso se usa la función "thingSpeakRead". Esta función toma como valores de entrada el número del canal, el campo y la contraseña del canal, ya que es un canal privado.

```
ModuleParameters=pvl_sapmmoduledb(181,"SandiaModuleDatabase_20120925.xlsx"); % Se define the PV module
Array.Ms = 2; %Número de módulos en serie
Array.Mp = 2; %Número de módulos en paralelo

time = (0:1:10);
% Se inicializan los variables de entrada
irradiancia = zeros ([1,11]);
Ee = zeros ([1,11]);
Tcel = zeros ([1,11]);
for i=1:length(time) % Guardamos los valores de irradiancia y temperatura en un array
  irradiancia(i) = thingSpeakRead(1060654,"Fields",6,ReadKey="HZN7A3PVOY3G2MTG");
  Ee(i) = irradiancia(i)/1000; % para tener la Ee en valores entre 0 - 1,
  Tcel(i) = thingSpeakRead(1060654,"Fields",1,ReadKey="HZN7A3PVOY3G2MTG");
end

mSAPMResults = pvl_sapm(ModuleParameters, Ee, Tcel);
aSAPMResults.Vmp = Array.Ms * mSAPMResults.Vmp;
aSAPMResults.Imp = Array.Mp * mSAPMResults.Imp;
aSAPMResults.Pmp = aSAPMResults.Vmp .* aSAPMResults.Imp;

%disp(aSAPMResults.Vmp) %muestra el valor de Vmp por pantalla
figure % Se muestra la irradiancia en función del tiempo
plot(time,irradiancia)

figure % Se muestra la temperatura en función del tiempo
plot(time,Tcel)


figure % Se muestra las caracterísitcas I - V de un panel solar fotovoltaico

subplot(3,1,1)
plot(time,aSAPMResults.Pmp,"-r")
legend("Pmp")
ylabel("Potencia DC (W)")
title("Características I - V")

subplot(3,1,2)
plot(time,aSAPMResults.Imp,"-r")
legend("Imp")
ylabel("Corriente DC (A)")

subplot(3,1,3)
```

```
plot(time,aSAPMResults.Vmp,"-r")
legend("Vmp")
ylabel("Voltaje DC (V)")
```

10.4. Código en Simulink con la entrada obtenida de ThingSpeak

El código desarrollado en este apartado se encarga de leer el valor de la irradiancia de ThingSpeak y enviárselo al modelo para que obtenga las salidas en función del dato obtenido.

Este código cuenta con tres partes importantes.

En la primera parte se halla la estructura "MemoryStruct", la cual está constituida por un puntero "char" para localizar el dato con el que se va a trabajar y un entero que determina el espacio que ocupa ese dato. Además, está la función "WriteMemoryCallback" que se encarga de reservar el espacio suficiente en la estructura definida para que el puntero guarde todos los datos y no se pierda ninguno.

En la segunda parte se encuentra la función "numero", que será la encargada de leer de ThingSpeak el valor de la irradiancia contenido en "field6". Para que se pueda obtener este dato constantemente hay que llamar a la función "número" dentro del while (1) que se encuentra en el main. Dentro de esta función se encuentran las siguientes variables:

- Un puntero llamado "curl" para buscar la información en la página web y la variable "res" en la que se almacena si se ha podido leer la información de la nube.
- Una estructura "MemoryStruct " la cual reserva espacio para poder guardar la información conseguida en "chunk" y se irá aumentando según sea necesario.
- Tres variables tipo "int" ("i", "j", "cont") y una variable tipo "char" ("aux"). La variable "i" se recorre toda la información guardada en "chunk" hasta encontrar el campo field6. Como aparecen varios "field6" cuando se obtiene toda la información de la nube, la variable "cont" se emplea para guardar solamente el segundo valor, el cual contiene el la irradiancia. Por último, la variable "j" recorre la cadena "aux" para almacenar en ella el valor obtenido de "field6".

Una vez definidas las variables que se van a usar se inicializa la función "CURL" con "curl_easy_init()" y se comprueba si se ha inicializado correctamente dentro del bucle "if". Dentro de este bucle se emplea la función "curl_easy_setopt()" para diferentes aplicaciones. La primera vez se indica a qué Localizador de Recursos Uniforme (URL) hay que conectarse. La segunda vez se utiliza para identificar si existen más direcciones URL y si las hay también tiene que obtener la información de ellas. La tercera vez, se llama a la función "WriteMemoryCallback", encargada de reserva memoria en la estructura creada al principio, en la que se guardan los datos obtenidos en la variable "chunk". Después de todo esto se comprueba si todo se ha ejecutado correctamente mirando el valor de "res", ya que si ha habido un fallo está variable es la encargada de mostrar un mensaje en el que indica que no se ha conseguido el dato deseado. Finalmente, se libera el "curl", la estructura y se devuelve el valor encontrado.

Por último, el desarrollo del main sigue el mismo funcionamiento que lo expuesto en los apartados 10.1 y 10.2, la única diferencia es que en este caso el valor de entrada hay que igualarlo a la función "numero".

Para poder implementar este código en la Raspberry Pi hay que ejecutar los comandos que se muestran en la figura 35. Pero en la sentencia en la que se compila el código, a parte del flag "lm" hay que añadir el flag "lcurl".

```
/*
 * Academic License - for use in teaching, academic research, and meeting
 * course requirements at degree granting institutions only.  Not for
 * government, commercial, or other organizational use.
 *
 * File: ert_main.c
 *
 * Code generated for Simulink model 'MPPT_CircuitoCelda'.
 *
 * Model version                  : 6.15
 * Simulink Coder version         : 9.7 (R2022a) 13-Nov-2021
 * C/C++ source code generated on : Sat Aug 20 12:31:42 2022
 *
 * Target selection: ert.tlc
 * Embedded hardware selection: ARM compatible-> ARM CORTEX-A
 * Code generation objectives:
 *    1. Execution efficiency
 *    2. RAM efficiency
 * Validation result: Not run
 */

#include <stddef.h>
#include <stdio.h>            /* This example main program uses printf/fflush */
#include "MPPT_CircuitoCelda.h"        /* Model header file */
#include <math.h>
#include <time.h>
#include <curl/curl.h>
#include <stdlib.h>


/* Este bloque define una estructura preparada para almacenar una memoria, que contiene
caracteres, y un tamaño. La usaremos para guardar la peticion */
struct MemoryStruct {
   char *memory;
   size_t size;
};

/* Esta es la funcion que salta cuando hemos terminado de recibir el contenido de la web y
estructura nuestra memoria conforme necesitemos alojar más datos */
static size_t
WriteMemoryCallback(void *contents, size_t size, size_t nmemb, void *userp)
{
   size_t realsize = size * nmemb;
   struct MemoryStruct *mem = (struct MemoryStruct *)userp;

   mem->memory = realloc(mem->memory, mem->size + realsize + 1);
   if(mem->memory == NULL) {
      /* out of memory! */
      printf("not enough memory (realloc returned NULL)\n");
      return 0;
   }

   memcpy(&(mem->memory[mem->size]), contents, realsize);
   mem->size += realsize;
```

```
    mem->memory[mem->size] = 0;

    return realsize;
}

float numero(void);
float numero(void){


    CURL *curl; //CURL es la biblioteca que permite conectarse a internet
    CURLcode res; //RES variable en la que se  almacena el estado de la peticion

    struct MemoryStruct chunk; /* Creamos una memoria de ese tipo llamado chunk */

    chunk.memory = malloc(1);  /* Iremos aumentando la memoria conforme necesitemos
espacio */
    chunk.size = 0;

    int i=0, j=0, cont=0; /* variables para encontrar field6 (irradiancia) */
    char aux[10];         /* valor encontrado de field6 */

    curl = curl_easy_init();  //Iniciamos curl con el método de la biblioteca.

    if(curl) { //Si hemos iniciado bien, procedemos

        //curl_easy_setop sirve para establecer configuracion a la hora de conectarse con CURL.
        //Aquí le decimos que queremos configurar la opción URL de curl. (Donde nos vamos a
conectar)
        curl_easy_setopt(curl, CURLOPT_URL,
"https://api.thingspeak.com/channels/1060654/feeds.json?api_key=HZN7A3PVOY3G2MTG&r
esults=2");
        //Le decimos que queremos seguir redirecciones, si las hubiese.
        curl_easy_setopt(curl, CURLOPT_FOLLOWLOCATION, 1L);
        //Le decimos que queremos que todo lo que devuelve la petición pase por la funcion
WriteMemoryCallback

        curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, WriteMemoryCallback);
        //Y los datos a guardar los tiene que escribir en nuestra estructura de memoria, llamada
chunk, por ello se la pasamos por referencia (&), por tal de que pueda modificarla.
        curl_easy_setopt(curl, CURLOPT_WRITEDATA, (void *)&chunk);

        //Lanzamos la peticion curl y almacenamos en res el estado final.
        res = curl_easy_perform(curl);

        //Si hay un error
        if(res != CURLE_OK){
            //Mostramos un mensaje diciendo por qué ha ido mal
            fprintf(stderr, "curl_easy_perform() falló: %s\n",
                    curl_easy_strerror(res));
        }/**else{
            //Si todo ha ido bien, podemos hacer cosas con lo que hay en nuestra memoria. En este
caso, mostrarla por pantalla:
            printf("Resultado: %s", chunk.memory);
        }*/
```

```
        while(1){
         if(chunk.memory[i]==102 && chunk.memory[i+5]=='6'){
          if(cont==1){
           i+=9;
           while(chunk.memory[i]!=34){
            aux[j] = chunk.memory[i];
            i++;
            j++;
           }
           break;
          }
          cont++;
         }
         i++;
        }

        //Liberamos y hacemos cleanup
        curl_easy_cleanup(curl);
        free(chunk.memory);
        return atof(aux);
    }
    return 0;
}

/*
 * Associating rt_OneStep with a real-time clock or interrupt service routine
 * is what makes the generated code "real-time".  The function rt_OneStep is
 * always associated with the base rate of the model.  Subrates are managed
 * by the base rate from inside the generated code.  Enabling/disabling
 * interrupts and floating point context switches are target specific.  This
 * example code indicates where these should take place relative to executing
 * the generated code step function.  Overrun behavior should be tailored to
 * your application needs.  This example simply sets an error status in the
 * real-time model and returns from rt_OneStep.
 */
void rt_OneStep(void);
void rt_OneStep(void)
{
  static boolean_T OverrunFlag = false;

  /* Disable interrupts here */

  /* Check for overrun */
  if (OverrunFlag) {
    rtmSetErrorStatus(rtM, "Overrun");
    return;
  }

  OverrunFlag = true;

  /* Save FPU context here (if necessary) */
  /* Re-enable timer or interrupt here */
  /* Set model inputs here */

  /* Step the model */
```

```
  MPPT_CircuitoCelda_step();

  /* Get model outputs here */

  /* Indicate task complete */
  OverrunFlag = false;

  /* Disable interrupts here */
  /* Restore FPU context here (if necessary) */
  /* Enable interrupts here */
}

/*
 * The example main function illustrates what is required by your
 * application code to initialize, execute, and terminate the generated code.
 * Attaching rt_OneStep to a real-time clock is target specific. This example
 * illustrates how you do this relative to initializing the model.
 */
int_T main(int_T argc, const char *argv[])
{
  /* Unused arguments */
  (void)(argc);
  (void)(argv);

  /* Initialize model */
  MPPT_CircuitoCelda_initialize();

  /* Simulating the model step behavior (in non real-time) to
   *  simulate model behavior at stop time.
   */
  double t0, t1;
  double tiempo;
  int a = 0;

  FILE *fichero;

  while (1) {
     fichero = fopen("aux.txt","a+");
     a = a + 1;
     t0 = clock();
     rtU.Inport = numero();
     rt_OneStep();
     t1 = clock();
     tiempo = (t1 - t0) / CLOCKS_PER_SEC;         //pasamos de tics a segundos

     fprintf(fichero,"%d irradiancia=%.2f\t", a, rtU.Inport);
     fprintf(fichero,"i_PV=%.4f\t", rtY.corriente);
     fprintf(fichero,"v_PV=%.4f\t", rtY.voltaje);
     fprintf(fichero,"p_PV=%.4f\t", rtY.potencia);
     fprintf(fichero,"tiempo = %lf s\n", tiempo);
     fclose(fichero);
  }

  return 0;
}
```

```
/*
 * File trailer for generated code.
 *
 * [EOF]
 */
```

11. Presupuesto

Costes de herramientas hardware

Se calcula como la suma del importe de cada elemento hardware utilizado, es decir, el ordenador de trabajo, la Raspberry Pi y todos los elementos que se conectan a ella (tabla 2).

- Raspberry PI 4 model B de 8GB de memoria RAM
- Fuente de alimentación con cable USB tipo C de 1.5m, para Europa, de color Negro
- Cable HDMI a micro HDMI Okdo de 2m de color Negro
- Tarjeta microSD de 32 GB
- Monitor Acer V196HQLAb 18.5" LED
- Owlotech MK100 Combo Teclado + Ratón Negro Portátil
- Portátil HP 15s-fq2021ns i7-1165G7/8GB/512SSD/15.6/W10

Equipo	Precio (€)	Duración	Amortización	Coste total (€)
Raspberry Pi	88,5	5 meses	4 años	9,22
Fuente de alimentación	9,18	5 meses	4 años	0,96
Cable HDMI	5,97	5 meses	5 años	0,50
Tarjeta microSD 32GB	6,6	5 meses	10 años	0,28
Monitor	96,67	5 meses	5 años	8,06
Teclado + ratón	5,99	5 meses	5 años	0,50
Portátil	649,99	7 meses	5 años	75,83
Total				95,33

Tabla 2: Costes hardware

Costes de herramientas software

Se calcula como la suma de la licencia de Matlab más el uso de Microsoft Office mostrado en la tabla 3.

Equipo	Precio (€)	Duración	Amortización	Coste total (€)
Matlab	840	7 meses	1 año	490
Windows Office	18,9	7 meses	Ilimitada	18,9
Total				508,9

Tabla 3: Costes software

Costes de mano de obra

Se requiere un ingeniero que desarrolle el gemelo digital y un escritor para que elabore la documentación del proyecto. Se trabaja un total de 620h, con 20h semanales durante 31 semanas. De los cuales, 6 meses ha estado el ingeniero desarrollando e implementando el proyecto y 1 mes el escritor ha redactado la documentación necesaria. Se muestran las horas y el coste por hora de cada uno en la tabla 4.

Personal	Horas trabajadas	Coste por h (€)	Total Coste (€)
Ingeniero	540	15	8100
Escritor	80	10	800
Total			8900

Tabla 4: Costes de mano de obra

Costes de ejecución material

Se obtienen como la suma del importe de los costes materiales (software y hardware) más los costes de la mano de obra (tabla 5).

Costes hardware	95,33
Costes software	508,90
Costes personal	8900
Total costes	9504,23

Tabla 5: Coste de ejecución material

Beneficio industrial

Se calcula el beneficio industrial de este proyecto como el 20% de los costes de ejecución material mostrado en la tabla 6.

Beneficio industrial	1900,85

Tabla 6: Beneficio industrial

Presupuesto de ejecución por contrata

Se logra como la suma del coste de ejecución material y del beneficio industrial (tabla 7).

Coste	Total (€)
Costes de ejecución	9504,23
Beneficio industrial	1900,85
Total	11405,08

Tabla 7: Presupuesto de ejecución por contrata

Honorarios

Según la ley 7/1997 del 14 de abril de medidas liberalizadoras en materia de suelo y Colegios Profesionales, se estima que los honorarios facultativos son un 10% del presupuesto total. En la tabla 8 se muestran los honorarios y el presupuesto total del proyecto.

Coste de ejecución por contrata	11405,08
Honorarios	1135,39
Total	12540,48

Tabla 8: Honorarios

Por lo tanto, el presupuesto final de este proyecto es de 12.540,48€.

12. Bibliografía

[1] L. Massel, N. Shchukin y A. Cybikov, “Digital twin development of a solar power plant” in E3S Web of Conferences 289, Energy Systems Research, 2021, pp. 6.

[2] Q. Li y Y. He, “An Overview of Digital Twin Concept for Key Components of Renewable Energy Systems”, International Journal of Robotics and Automation Technology, vol. 8, pp. 29 – 47, 2021.

[3] P. Palensky, M. Cvetkovic, D. Gusain y A. Joseph, “Digital twins and their use in future power systems” Digital Twin, vol. 1, pp. 14, septiembre 2021.

[4] Capgemini (30 mayo, 2022). Digital Twins: Adding Intelligence to the Real World [Online]. Disponible: https://www.capgemini.com/es-es/resources/gemelos-digitales-la-inteligencia-artificial-adaptada-al-mundo-real/

[5] Wikipedia (5 junio, 2022) Célula fotoeléctrica [Online]. Disponible: https://es.wikipedia.org/wiki/C%C3%A9lula_fotoel%C3%A9ctrica#Principio_de_funcionamiento

[6] A. G. Vera-Dávila, J. C. Delgado- Ariza y S. B. Sepúlveda-Mora, (2018). “Validación del modelo matemático de un panel solar empleando la herramienta Simulink de Matlab” Revista de Investigación, Desarrollo e Innovación. vol. 8, no. 2, pp. 343-356, 2018.

[7] O. Planas (25 febrero, 2016). Partes de un panel solar, componentes y estructura [Online]. Disponible: https://solar-energia.net/energia-solar-fotovoltaica/elementos/panel-fotovoltaico/estructura-de-un-panel-fotovoltaico#

[8] A. J. Fuentes Tinoco y R. D. López Reyes, Revisión de algoritmo para el seguimiento del punto de máxima potencia de un panel solar, informe final de trabajo de grado, Santander, Unidades tecnológicas de Santander, Bucaramanga, Colombia, 2021.

[9] "Photovoltaic Solar Energy", class notes for 370014, Departamento de electrónica, Universidad de Alcalá de Henares, 2018.

[10] P. Sánchez Molina (26 junio, 2019). España, el segundo mercado fotovoltaico de Europa durante los próximos 5 años [Online]. Disponible: https://www.pv-magazine.es/2019/06/26/espana-el-segundo-mercado-fotovoltaico-de-europa-durante-los-proximos-5-anos/

[11] Solectroshop (24 abril, 2021). Convertidores Buck/Boost – Step up/Step Down [Online]. Disponible: https://solectroshop.com/es/blog/convertidores-buckboost-step-upstep-down-n82

[12] Wikipedia (22 agosto, 2022). Socket de Internet [Online]. Disponible: https://es.wikipedia.org/wiki/Socket_de_Internet

[13] The MathWorks, Inc. (1994 - 2022). Matlab [Online]. Disponible: https://www.mathworks.com/products/matlab.html

[14] The MathWorks, Inc. (1994 - 2022). Simulink [Online]. Disponible: https://es.mathworks.com/products/simulink.html

[15] PVPerformance Modeling Collaborative. PV_LIB Caja de herramientas para documentación y ayuda de funciones de Matlab [Online]. Disponible: https://pvpmc.sandia.gov/PVLIB_Matlab_Help/

[16] The MathWorks, Inc. (2022) ThingSpeak [Online]. Disponible: https://thingspeak.com/

[17] European Commission. PVGIS Photovoltaic Geographical Information System [Online]. Disponible: https://joint-research-centre.ec.europa.eu/pvgis-photovoltaic-geographical-information-system_en

[18] Wikipedia (11 agosto, 2022). VNC [Online]. Disponible: https://es.wikipedia.org/wiki/VNC#:~:text=VNC%20son%20las%20siglas%20en,trav%C3%A9s%20de%20un%20ordenador%20cliente.

[19] E. Pons. "¿QUÉ ES? TUTORIAL COMPLETO en ESPAÑOL de como configurar la Raspberry Pi 4 B+ CURSO completo". (2019, mayo 22). [Video en línea]. Disponible: https://www.youtube.com/watch?v=GXUncVcLpFk

[20] Raspberry Pi Tranding Ltd. (junio, 2019). Raspberry Pi 4 Computer Model B [Online]. Disponible: https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-4-Product-Brief.pdf

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR