



Universidad
Zaragoza

Trabajo Fin de Grado

Diseño y desarrollo de sistema de gestión energética
en tiempo real para automatización domótica

Design and development of a real-time energy
management system for home automation

Autor

Diego Remón Blesa

Director

Roberto Casas Nebra
Álvaro Marco Marco

ESCUELA DE INGENIERÍA Y ARQUITECTURA
2023

AGRADECIMIENTOS

En primer lugar, me gustaría expresar mi agradecimiento a mis tutores, Roberto Casas Nebra y Álvaro Marco Marco, por la orientación y supervisión de este proyecto como es debido. Su dirección y enfoque preciso han sido esenciales para dar forma este trabajo de manera adecuada. A lo largo de este tiempo me han guiado, permitiéndome aprender nuevas habilidades que no solo han sido fundamentales en el trabajo, sino que también se extienden a mi futura carrera profesional.

También deseo extender mi reconocimiento a investigadores y desarrolladores que con su esfuerzo, dedicación y trabajo contribuyeron a enriquecer mi investigación en el ámbito de este proyecto. Su continua búsqueda de avances y descubrimientos ha sido una muy importante fuente de información y conocimiento a la hora de llevar a cabo este trabajo.

Por último, pero no por ello menos importante, quiero expresar mi más especial gratitud a mi padre, a mi madre y al resto de mi familia y amigos. Su constante apoyo moral y confianza resultaron ser pilares fundamentales en este proceso, obteniendo respaldo muy importante en momentos difíciles o desafiantes que me proporcionó la fuerza para superarlos y seguir adelante.

RESUMEN

En la actualidad, la concienciación con el uso de energías renovables, necesidad de un menor consumo que conlleve a gastos inferiores y el deseo de un bienestar lo más continuo y óptimo posible hacen que sea necesario encontrar una o distintas formas de conseguir solucionar todo ello. Por lo tanto, se busca reducir el consumo de energía o mantener uno similar de manera más sostenible y adecuado, para que esto refleje un beneficio en nuestro día a día. A su vez, el bienestar es un objetivo importante en nuestra vida, siendo por ello esencial la investigación y desarrollo de algún método o sistema que consiga un mayor confort.

Este Trabajo de Fin de Grado (*TFG*) busca encontrar una forma de poder gestionar un entorno en el que hay un consumo energético, con la intención de que sea lo más óptimo posible, y teniendo como objetivo mediante distintos tipos de automatizaciones domóticas un mayor bienestar en nuestro día a día.

Para ello, se tiene como objetivo el diseño de un sistema basado en un software y un hardware que permitan la gestión y control del flujo de energía para una eficiente automatización domótica en tiempo real. Utilizando para ello distintos tipos protocolos de comunicación que faciliten adquirir de manera efectiva los datos necesarios para lograr un control adecuado del entorno. Por tanto, se busca una estructura de hardware que habilite el desarrollo del software sobre este, permitiendo ejecutarlo de manera correcta y teniendo en cuenta que su arquitectura sea dual facilitando mediante un procesador en tiempo real tratar datos del entorno y ejecutar órdenes adecuadamente.

En primer lugar, se explicará el entorno a desarrollar las distintas características que lo componen y cual será la metodología a seguir para poder solucionarlo. A continuación, se tomará la decisión comparando distintos softwares y diferentes hardwares de cual es el más idóneo para solventar el problema. Con un software y un hardware ya decididos, se procederá a explicar como funcionan cada uno de ellos y a su vez, se analizará y desarrollará el diseño de ambos comentando las tomas de decisiones y el procedimiento realizado.

Por tanto, a lo largo de este trabajo se desarrollan las distintas fases mencionadas, llevando a cabo distintas tomas de decisiones sobre el diseño y desarrollo del software, y del hardware sobre el que se va a ejecutar. Se aclararán distintos procedimientos realizados y resultados obtenidos, tomando finalmente una serie de conclusiones y mencionando posibles líneas futuras.

Índice general

Índice de figuras	VI
Índice de cuadros	VIII
1. Introducción	1
1.1. Motivación y contexto	1
1.2. Objetivos	2
1.3. Alcance	2
1.4. Trabajos similares	3
1.5. Estructura de la memoria	3
2. Estado del arte	4
2.1. Introducción	4
2.2. Entorno a controlar	5
2.2.1. Red Eléctrica	5
2.2.2. Paneles Fotovoltáicos	6
2.2.3. Red de Sensores	7
2.2.4. Usuarios y actuación propia	8
2.2.5. Representación del espacio	8
2.3. Protocolos de comunicación	9
2.3.1. Protocolos industriales	9
2.3.2. Protocolos domésticos	9
2.3.3. Comunicación con la nube	10
2.4. Necesidades de Software	10
2.5. Necesidades de Hardware	10
2.6. Tablas comparativas	10
2.7. Conclusiones y decisiones	15
3. Hardware	16
3.1. Introducción	16
3.2. Arquitectura del Hardware	16
3.2.1. System on Module	16
3.2.2. Single Board Computer	17
3.2.3. Funcionamiento y uso	18
3.3. Diseño del Hardware	19
3.3.1. Comenzando con el módulo	19
3.3.2. Yocto OS: tests y uso de la interfaz	20
3.3.3. Instalación de Debian	21

3.3.4.	Sensores	22
3.3.5.	Módulo de entradas/salidas digitales	23
3.4.	Co-Procesador en tiempo real Cortex-M33	24
3.4.1.	Puesta en marcha del co-procesador	24
3.4.2.	Aplicación del co-procesador en tiempo real	24
4.	Software	27
4.1.	Introducción	27
4.2.	Arquitectura del Software	27
4.2.1.	Definición del Software	27
4.2.2.	Características del Software	27
4.2.3.	Instalación del Software	28
4.2.4.	Funcionamiento y uso del Software	29
4.3.	Diseño del Software	30
4.3.1.	Instalación de Home Assistant	30
4.3.2.	Comenzando con Home Assistant	32
4.3.3.	Interfaz y ajustes de Home Assistant	32
4.3.4.	Configuración de sensores, integraciones y comunicaciones . . .	33
4.3.5.	Diseño de automatizaciones	37
5.	Análisis de la información obtenida a partir de los resultados	42
5.1.	Introducción	42
5.2.	Resultados y características de las automatizaciones	42
5.3.	Resultados del análisis del inversor Modbus	43
6.	Conclusiones y líneas futuras	46
6.1.	Introducción	46
6.2.	Conclusiones	46
6.3.	Líneas futuras	47
A.	Anexos	48
A.1.	Anexo 1: Análisis detallado de las alternativas de software	48
A.1.1.	Objetivos y características del software	48
A.1.2.	OpenHAB	49
A.1.3.	Domoticz	50
A.1.4.	RexControls	51
A.1.5.	Codesys	52
A.1.6.	OpenRemote	52
A.1.7.	EdgeXFoundry	53
A.1.8.	ROS	54
A.1.9.	Home Assistant	55
A.1.10.	Microsoft Azure IoT Edge	57
A.1.11.	Node-RED	58
A.1.12.	Homebridge	59
A.2.	Anexo 2: Análisis detallado de las alternativas de hardware	62
A.2.1.	Arm	63
A.2.2.	Intel	64
A.2.3.	Microchip	65
A.2.4.	NXP	66

A.2.5. Qualcomm	69
A.3. Anexo 3: Puesta en marcha del módulo VAR-SOM-MX93	73
A.3.1. Yocto OS: tests y uso de la interfaz	74
A.3.2. Instalación de Debian	77
A.4. Anexo 4: Instalación y puesta en marcha de Home Assistant	78
A.5. Anexo 5: Instalación de broker mosquitto y configuración en Dingtian .	80
A.6. Anexo 6: Instalación de MCUXpresso y aplicación detallada	81
B. Bibliografía	83

Índice de figuras

2.1. Esquema Calentador Solar Fotovoltáico (Fuente: Hogarsense)	6
2.2. Aprovechamiento de la energía eléctrica en forma de Corriente Continua	7
2.3. Entorno a controlar	8
3.1. VAR-SOM-MX93 (Fuente: Variscite)	17
3.2. Symphony Board (Fuente: Variscite)	18
3.3. Comunicación con WiFi	21
3.4. Dongle Plus (Fuente: SONOFF)	22
3.5. Smart Plug (Fuente: SONOFF)	22
3.6. SNZB-01 (Fuente: Amazon)	22
3.7. SNZB-04 (Fuente: SONOFF)	23
3.8. Motion (Fuente: Xiaomipedia)	23
3.9. Sensor T ^a (Fuente: SONOFF)	23
3.10. 16 Channel Relay Board (Fuente: dingtian-tech)	23
3.11. Activación de una salida a partir de una entrada con M-33	25
4.1. Logotipo de Home Assistant	27
4.2. Mejores softwares gratuitos para domótica (Fuente: LinuxLinks)	28
4.3. Comparativa de métodos de instalación (Fuente: Home Assistant)	29
4.4. Configuración inicial de Home Assistant	32
4.5. Tarjeta de control y visualización del entorno domótico	32
4.6. Integración de Zigbee Home Automation y ZBDongle-E	33
4.7. Uso de sensores Zigbee tras su configuración	34
4.8. Comunicación MQTT entre el módulo de relés y Home Assistant	35
4.9. Activación y lectura de relés en Home Assistant	35
4.10. Configuración Modbus de un inversor fotovoltaico	36
4.11. Configuración de otros dispositivos modbus	37
4.12. Acceso a Home Assistant desde un dispositivo móvil	37
4.13. Automatización de ejemplo del switch	38
4.14. Automatizaciones de distintos dispositivos climatizadores	39
4.15. Automatización de luces dependiendo del pulso del pulsador Zigbee	40
4.16. Automatización de apertura/cierre de una puerta	40
4.17. Automatización de la puerta mediante editor visual	41
5.1. Comparación de recursos usados en la automatización del ventilador	42
5.2. Resultados de la automatizaciones en un mapa de Home Assistant	43
5.3. Tensión del campo A (V)	43
5.4. Corriente del campo A (A)	44
5.5. Potencia del campo A (W)	44

5.6. Potencia de la batería (W)	45
5.7. Porcentaje de carga de la batería (%)	45
A.1. Logotipo de OpenHAB	49
A.2. Logotipo de Domoticz	50
A.3. Logotipo de RexControls	51
A.4. Logotipo de Codesys	52
A.5. Logotipo de OpenRemote	53
A.6. Logotipo de EdgeXFoundry	54
A.7. Logotipo de ROS	55
A.8. Logotipo de Home Assistant	56
A.9. Logotipo de Microsoft Azure IoT Edge	57
A.10. Logotipo de Node-RED	58
A.11. Logotipo de Homebridge	59
A.12. Rapidez de comercialización respecto a su grado de integración	62
A.13. Logotipo de Arm	63
A.14. Proyecto de Arm basado en los estándares, ecosistemas y seguridad	63
A.15. Logotipo de Intel	64
A.16. Intel NUC Boards and Elements	64
A.17. Logotipo de Microchip	65
A.18. Logotipo de NXP	66
A.19. Estructura de un procesador NXP (primera parte)	67
A.20. Estructura de un procesador NXP (segunda parte)	68
A.21. Logotipo de Qualcomm	69
A.22. Configuración del host Putty	73
A.23. Mensajes recibidos mediante comando <i>sudo dmesg</i> en Ubuntu	74
A.24. Cores activos de la CPU	75
A.25. Temperatura de la CPU medida en miligrados Celsius	76
A.26. Información de la comunicación Ethernet del módulo	76
A.27. Comunicación Ethernet test TCP	76
A.28. Comunicación con WiFi	77
A.29. Arquitectura del kernel Linux instalado	79
A.30. Estado del broker mosquitto tras la instalación	80
A.31. Configuración del broker mosquitto en la interfaz del módulo de relés	80

Índice de cuadros

2.1 Comparativa de Softwares 1	11
2.2 Comparativa de Softwares 2	12
2.3 Comparativa Hardware (System on Modules)	13
2.4 Comparativa Hardware (Single Board Computers)	14
2.5 Comparativa Hardware (Otras SBCs y Kits de desarrollo)	14
A.1 Decision Software a utilizar extendido	61
A.1 Decision Hardware a utilizar extendido (SoMs)	71
A.1 Decision Hardware a utilizar extendido (SBCs/Kits de desarrollo)	72

Capítulo 1

Introducción

1.1. Motivación y contexto

El consumo energético y el bienestar son características muy importantes en la actualidad, buscando que cualquier persona pueda vivir de la mejor manera posible, con las mayores facilidades a la hora de afrontar el día a día, y a su vez, realizar un consumo energético de la manera más eficiente. Todo ello, con el objetivo de que al final del día, se pueda aprovechar el mismo nivel de energía con menores gastos, con ello un mejor precio y de esta forma el consumidor se encuentre más satisfecho.

Actualmente el consumo energético es muy relevante en la sociedad, llevado de la mano de la revolución de las energías renovables. Por ello, un empleo eficiente y adecuado de las energías que entran en un determinado espacio dando prioridad a las renovables, como puede ser la energía solar, es determinante en el presente para poder realizar un consumo más responsable y agradable con el medio ambiente.

Por otro lado, el bienestar de las personas se ve reflejado en estos tiempos de cierta forma según el grado de automatización del entorno, logrando con un lugar automatizado correctamente y de manera agradable satisfacer al usuario.

De la misma forma se encuentra notoriamente implementado en la sociedad un alto uso de tecnologías inalámbricas, tanto en ámbito sensorial como con redes de comunicación que se aprecian de manera global, como pueden ser WiFi, Bluetooth o MQTT entre otros muchos, desarrollando una conexión colectiva de distintos dispositivos. Con ello se forma lo que se conoce como IoT (Internet of Things), que se encuentra en continua evolución y altamente utilizado actualmente en cuanto a dispositivos inteligentes.

Por todo esto, este Trabajo de Fin de Grado tiene como objetivo conseguir abarcar estos ámbitos ya nombrados, logrando que en una única solución se pueda gestionar y controlar de la manera más eficiente posible un entorno que tiene como entrada energías de distintos tipos, con el fin de facilitar al usuario que se encuentre en este determinado entorno su bienestar y alcanzar un consumo más favorable. Todo esto podrá ser útil ya no tan solo en lo que es una vivienda de uso diario, sino también a nivel de edificio o en ámbito industrial.

1.2. Objetivos

El objetivo principal de este Trabajo de Fin de Grado es el diseño y desarrollo de un sistema que consiga gestionar de la manera más eficiente posible el flujo de energía en un determinado lugar para su automatización domótica en tiempo real, utilizando para ello distintos protocolos de comunicación que permitan adquirir de manera efectiva los datos necesarios y así lograr un control adecuado del entorno.

Para lograr este objetivo principal se abordarán los siguientes objetivos:

- Elección de un software que ejecutado en tiempo real pueda controlar todo el sistema de la mejor manera posible, permitiendo las comunicaciones necesarias para que esto se cumpla, que pueda funcionar en distintas situaciones como de forma remota, offline u online, comunicarse con la nube y que muestre una interfaz amigable para su uso y control del usuario.
- Diseño y desarrollo de este software, logrando gestionar de manera correcta el entorno que se desea controlar.
- Encontrar un hardware que permita ejecutar este software en tiempo real de manera correcta y facilite las distintas comunicaciones a realizar adecuadamente.
- Ejecutar el software sobre el hardware y comprobar que el sistema funciona de la manera deseada analizando los resultados, consiguiendo así un control energético de la forma más eficiente posible.

1.3. Alcance

Siguiendo el objetivo principal de este trabajo, se organiza el problema de manera que se cumplan el resto de los objetivos que lo constituyen, descomponiéndolo en distintas fases.

En primer lugar se analizarán una serie de softwares cuyo funcionamiento sirva para automatización domótica e IoT, y de esta forma, encontrar el más óptimo para la consecución del objetivo y la facilidad de su uso.

A continuación, se buscará entre distintos proveedores el mejor o más adecuado hardware, sobre el que poder ejecutar correctamente el software que se ha elegido, teniendo en cuenta que pueda realizar órdenes en tiempo real.

En base a lo primero, se procederá a diseñar y desarrollar el software consiguiendo que este sea lo más eficiente y óptimo posible.

Con el hardware presente, se instalará el sistema operativo necesario, se harán las pruebas necesarias y se comprobará que el hardware funciona adecuadamente.

Con el código y software diseñado, se ejecutará el software sobre el hardware escogido y con ello se realiza la comprobación de que todo funciona correctamente.

Posteriormente, se efectúa un análisis de los resultados obtenidos, comprobando que el comportamiento de las distintas características relevantes del trabajo se cumplen.

Finalmente, se extraen las conclusiones en base a los resultados obtenidos de la ejecución del software y se exponen posibles líneas futuras.

1.4. Trabajos similares

Algunos estudios y trabajos relacionados a los que se enfrenta este son:

- **Design and Implementation of a Real-Time Smart Home Management System Considering Energy Saving[1].**

Este trabajo tiene como principal objetivo la ayuda a los consumidores a controlar sus electrodomésticos para asistir a personas mayores y discapacitadas. Además se propone una forma eficiente de gestionar la energía dentro de una vivienda aislada, protegiendo baterías de sobrecargas y buscando menor coste de funcionamiento. Se logran buenos resultados tras usar distintos tipos de comunicación IoT y una Raspberry PI, además de poder monitorizar todos los sensores a través de una aplicación móvil.

- **Design of Smart Home Energy Management System for Saving Energy [2].**

En este trabajo se propone un sistema de gestión energética que utiliza la tecnología WiFi para transmisión de datos dentro de casa, y comunicación GSM (Global Mobile System) de manera externa, consiguiendo llegar a un ahorro energético del 18 % en las pruebas realizadas. El empleo de este sistema de gestión enriquece el ahorro energético sin tener que añadir equipos caros, proporcionando información a los residentes para que puedan controlar el hogar a través de SMS.

1.5. Estructura de la memoria

El contenido a realizar en este trabajo se resume de la siguiente manera:

- **Capítulo 2 - Estado del Arte:** en este capítulo se estudian los tipos de energía que acceden a un determinado edificio/vivienda, sus aplicaciones y como conocer sus valores. Estas generarán unas necesidades que servirán para analizar distintos softwares y hardwares, escogiendo finalmente los más adecuados para lograr los objetivos.
- **Capítulo 3 - Software:** en este capítulo se analiza la arquitectura del software escogido apreciando como funciona, su uso y características. Con ello se procederá a diseñar el software que se ejecutará posteriormente.
- **Capítulo 4 - Hardware:** en este capítulo se analiza la arquitectura del hardware escogido mostrando sus principales características. Posteriormente, se procederá a explicar los pasos realizados para la puesta a punto del hardware y con ello se pueda ejecutar el software correctamente.
- **Capítulo 5 - Análisis de la información obtenida a partir de los resultados:** capítulo en el que se recogen y exponen los resultados obtenidos.
- **Capítulo 6 - Conclusiones:** en este capítulo se exponen las conclusiones obtenidas y se indican ciertas líneas futuras.

Capítulo 2

Estado del arte

Para la realización de este capítulo se ha buscado información sobre distintos tipos de energías, sus efectos e importacia en un edificio/vivienda, protocolos de comunicación tanto industriales como domésticos y se han analizado distintas páginas webs de diferentes softwares y hardwares para su exhaustiva comparación y posterior elección de los más adecuados para el proposito a lograr.

2.1. Introducción

Debido a un aumento de manera exponencial, conforme pasa el tiempo de la población humana, los recursos energéticos son cada vez más escasos. Con los métodos de consumo que se han estado aplicando las últimas décadas, la gran parte de la energía generada se desperdicia en la red de distribución, por lo que se ha estado buscando una red inteligente que solucione los problemas energéticos del mundo moderno[3].

Los hogares actualmente necesitan un determinada cantidad de energía para alimentar distintos dispositivos y equipos domésticos, teniendo consumos elevados de diferentes tipos de energía que han ido aumentando con el tiempo entre las que destacan sobretudo la electricidad y el gas natural.

El uso de estas fuentes de energía sirve principalmente para suplir las necesidades básicas del día a día como son la luz, calefacción, el uso de agua caliente o para la cocina[4].

Sin embargo, el avance respecto al uso de energías renovables y la continua necesidad de realizar el cambio a ellas ha hecho que aparezcan nuevas formas de aprovechar la energía en una construcción, con el objetivo de tener mismos consumos a un precio más asequible y reducir el impacto medioambiental. Entre ellas, destacan sobretudo en nuestro país la energía solar captada mediante paneles fotovoltaicos o la energía eólica. No obstante, también se encuentran otras como son la hidráulica o la biomasa, que consiguen generar energía a partir de la potencia del agua o de la combustión de materia orgánica respectivamente.

La energía solar es la que se tendrá en cuenta para este trabajo ya que entre las energías renovables, esta es la que suele prevalecer en la construcción[5]. Esto se debe a que es una energía que como se indica por definición, es inagotable. Además, por la situación de nuestro planeta, es abundante. Llega a todas las partes del planeta

pudiéndose aprovechar en mayor o menor medida. Por otro lado, se combina muy bien con baterías y red eléctrica lo que lo hace muy interesante como complemento para reducir consumos, es versátil, pudiendo además de generar electricidad, calentar determinados fluidos, siendo escasa la necesidad de su mantenimiento y soportando una tecnología sólida y confiable[6].

Para ello, se van a necesitar distintos protocolos de comunicación que van a ser analizados posteriormente, y con los que se podrán conocer los datos a extraer y poder controlar el entorno de la manera más óptima. Estos datos se deseará que sean controlados mediante un software que recoja las características más idóneas para cubrir todas las especificaciones y una única solución hardware que permita de forma cómoda analizar y gestionar el entorno en tiempo real.

2.2. Entorno a controlar

Como se ha nombrado en la introducción, se va a tratar una construcción o entorno en el que entran distintos tipos de energía y que tendrán que ser controlados de la mejor manera para optimizar su consumo y conseguir mejores resultados. En este trabajo se va a centrar en el aprovechamiento de la energía eléctrica tanto por parte de la *Red Eléctrica*, como de *Paneles Fotovoltáicos* a partir de energía solar.

Se va a utilizar por tanto un sistema embebido con el que se controlarán los distintos dispositivos de la construcción que aporten la información sobre este lugar, y permitirá activar las salidas adecuadas dependiendo del estado del entorno en cada momento.

Para ello, se analizarán los datos obtenidos de distintas fuentes como son la red eléctrica, paneles solares, sensores en el hogar, nube, entradas y salidas digitales..., que permitirán facilitar la decisión en cada momento de cual es gestión adecuada.

2.2.1. Red Eléctrica

En primer lugar se encuentra la Red Eléctrica, esta es la fuente principal en la mayoría de viviendas e industria actualmente, consumiendo según la **REE** (*Red Eléctrica Española*), unos 9 kWh de media diarios, que serían unos 270kWh al mes, con un coste que ronda los 50-60€ dependiendo de la distribución horaria o del tipo de vivienda entre otros[7].

La Red Eléctrica alimentará distintos dispositivos importantes en la vivienda como son el **Climatizador o Aire Acondicionado**. Este toma *Corriente Alterna*, debido a que estos dispositivos contarán con compresores y motores que se alimentan con este tipo de corriente.[8]. A partir de estos climatizadores se tomará cierta información, y se activarán o no mediante distintos protocolos de comunicación; destaca sobretodo *Modbus*, aunque se podrían utilizar otros como *BACnet* o el estandar *KNX*.[9]

Por otro lado la vivienda contará con un *Contador* de luz que se encargará de registrar el consumo efectuado y conocer el estado de la red, por lo tanto, también será alimentado por la Red Eléctrica[10]. De igual forma que ocurría con los climatizadores, habrá distintas posibilidades respecto al protocolo de comunicación que se utilice. Sin embargo, destaca de manera notable el protocolo industrial *Modbus*, siendo de los más usados en el mundo en cuanto a este tipo de dispositivos.[11]

Respecto al agua caliente, se dispone en gran cantidad de instalaciones de diferentes *Calentadores Eléctricos de Agua*, estos se suelen alimentar desde la red eléctrica, con el objetivo de alimentar el sistema que lo compone y calentar el agua[12]; no obstante, hay otros métodos de calentar el agua mediante calentadores a partir de paneles solares que se señalarán posteriormente. Los calentadores AC permiten varios protocolos de comunicación, entre los que destacan hoy en día métodos inalámbricos como WiFi o Zigbee[13], aunque también podrían utilizarse otros como Modbus[14].

2.2.2. Paneles Fotovoltáicos

Los paneles solares son la instalación encargada de absorber la radiación del sol, fuente de energía renovable e ilimitada, y con la que genera la energía eléctrica. La energía solar se genera de dos formas, mediante captores térmicos o células fotovoltaicas compuestas de silicio.

A día de hoy hay gran oferta de placas solares y se pueden encontrar a un precio más que asequible. A su vez, los kits solares aportan un amplio abanico de posibilidades para gestión de energía limpia y renovable[15]. Se podrá generar *Corriente Continua (DC)* a partir de la energía solar, que se podrá utilizar para distintas aplicaciones.

Cabe mencionar la existencia de *Calentadores Solares Fotovoltáicos*, con los que a partir de la energía solar captada se podrá obtener el agua caliente necesaria para su uso en distintos ámbitos como en **ACS** (*Agua Caliente Sanitaria*) o en **Calefacción**, actuando sobre la temperatura del agua para su consumo, como ajustando la temperatura del entorno con la calefacción.

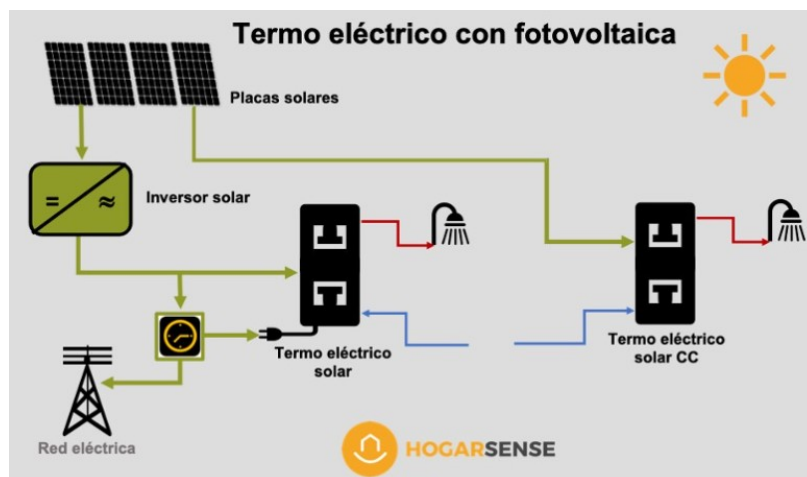


Figura 2.1: Esquema Calentador Solar Fotovoltáico (Fuente: Hogarsense)

Estos calentadores solares fotovoltaicos son equipos que utilizan energía solar térmica o fotovoltaica. Sirven para generar ACS o calefacción en invierno y si están cubiertas las necesidades, destinar la electricidad a otros ámbitos; en el caso de combinarse con una bomba de calor, también se podrá climatizar en verano, que es más rentable y ecológico. Esto le da ventaja frente al calentador eléctrico nombrado previamente, ya que se podrá unir en un mismo dispositivo la capacidad de generar calor en invierno, y climatizar cuando la temperatura sube[16]. Estos calentadores permiten la comunicación mediante distintos protocolos como pueden ser Wifi, MQTT o HTTP[17], aunque también se suele utilizar de manera habitual el protocolo Modbus[18].

La energía solar recibida y convertida en energía eléctrica en forma de *Corriente Continua (DC)*, puede utilizarse en primer lugar para alimentar la vivienda en combinación con la energía de la red eléctrica, incluso pudiendo devolver la energía sobrante a la red, al convertir esta corriente DC en *Corriente Alterna (AC)* mediante un *Inversor solar o fotovoltaico*[19]. Estos inversores serán controlados mediante protocolos de comunicación que podrían ser inalámbricos como WiFi o Zigbee entre otros[20]; sin embargo, principalmente se utiliza Modbus para la obtención de datos y su control[21].

La corriente DC obtenida podrá usarse, si es necesario, para almacenar la energía en baterías o cargarlas, teniendo en cuenta la capacidad de los paneles solares y de las baterías para posteriormente realizar las conexiones correctamente mediante un cargador de baterías[22]. Este sistema de carga sirve también en el caso de necesitar cargar un vehículo propio, pudiendo realizar una carga parcial teniendo los paneles solares, o aprovechando la carga almacenada en las previas baterías en el caso de estar en un día con ausencia o reducida luz solar[23]. Para el uso de estos cargadores, su control y análisis se puede realizar mediante diferentes protocolos de comunicación, siendo de los más habituales *CAN Bus*[24].

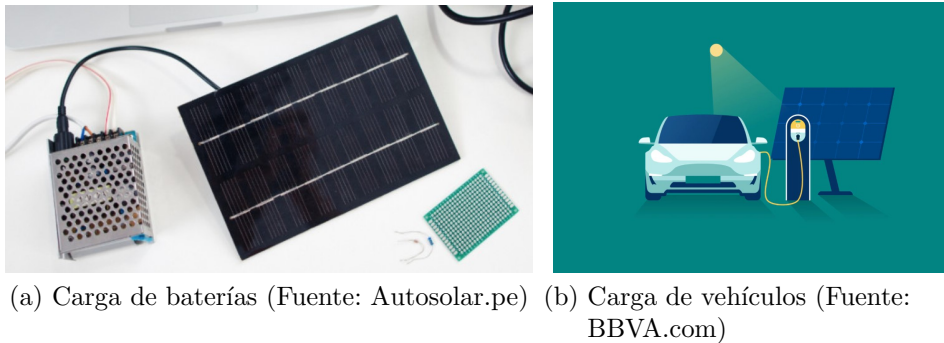


Figura 2.2: Aprovechamiento de la energía eléctrica en forma de Corriente Continua

2.2.3. Red de Sensores

A parte de la energía eléctrica que habrá que gestionar, el entorno será un lugar *inteligente* y sensorizado adecuadamente mediante distintos tipos de sensores (temperatura, movimiento, humedad...), por lo que se necesitará de métodos inalámbricos e IoT que harán que la automatización de la casa sea lo más precisa posible.

El **IoT** (*Internet of Things*), es el proceso que permite conectar los elementos cotidianos a Internet, comunes como bombillas, recursos para salud, accesorios inteligentes o sistemas de las ciudades inteligentes. En este ámbito se dividen en dos categorías, interruptores actuando sobre objetos, o sensores que recopilan datos y los envían a otro lugar, todo ello a través de redes inalámbricas con la mínima intervención humana, desarrollando tecnologías en auge como son, por ejemplo, los sistemas de inteligencia artificial y aprendizaje automático (IA/ML).[25]

Por lo tanto, el entorno necesitará de distintos métodos de comunicación doméstica e IoT con los que poder controlarlo. Principalmente y de manera típica, se pueden encontrar protocolos como WiFi, Zigbee, Bluetooth/BLE o MQTT que permiten enviar

y recibir información de los sensores y sistemas inteligentes. A su vez, será necesario el acceso a la nube, que permitirá conocer información relevante que favorezca tanto en el ámbito energético como para una mejor gestión de los datos obtenidos sensorialmente.

2.2.4. Usuarios y actuación propia

Los individuos que se encuentren en este entorno también aportarán información y pueden actuar de forma autónoma. Se podrá por tanto comunicarse vía IP con los distintos dispositivos de cada persona, que aportarán su identificación, localización, o tiempo que permanecen en un lugar, entre otras características. Esto permitirá gestionar según gustos y hábitos de los usuarios. Entre estos dispositivos se pueden encontrar móviles, tablets u ordenadores que permitan identificar a distintos usuarios al acceder a la red local del sistema.

Por otra parte, estos usuarios puede ser que necesiten activar, desactivar, encender o apagar distintos dispositivos manualmente, por tanto habrá distintos interruptores, pulsadores, activadores, etc. Estos permitirán controlar el entorno de manera autónoma si los usuarios lo ven oportuno.

2.2.5. Representación del espacio

Con todo lo analizado, se necesitará un sistema que permita controlar y gestionar las distintas entradas de la energía que llega al entorno, como también conocer y manejar adecuadamente la inteligencia del lugar mediante su sistema de sensores y los usuarios que lo habitan. Esto se puede representar en el siguiente esquema:

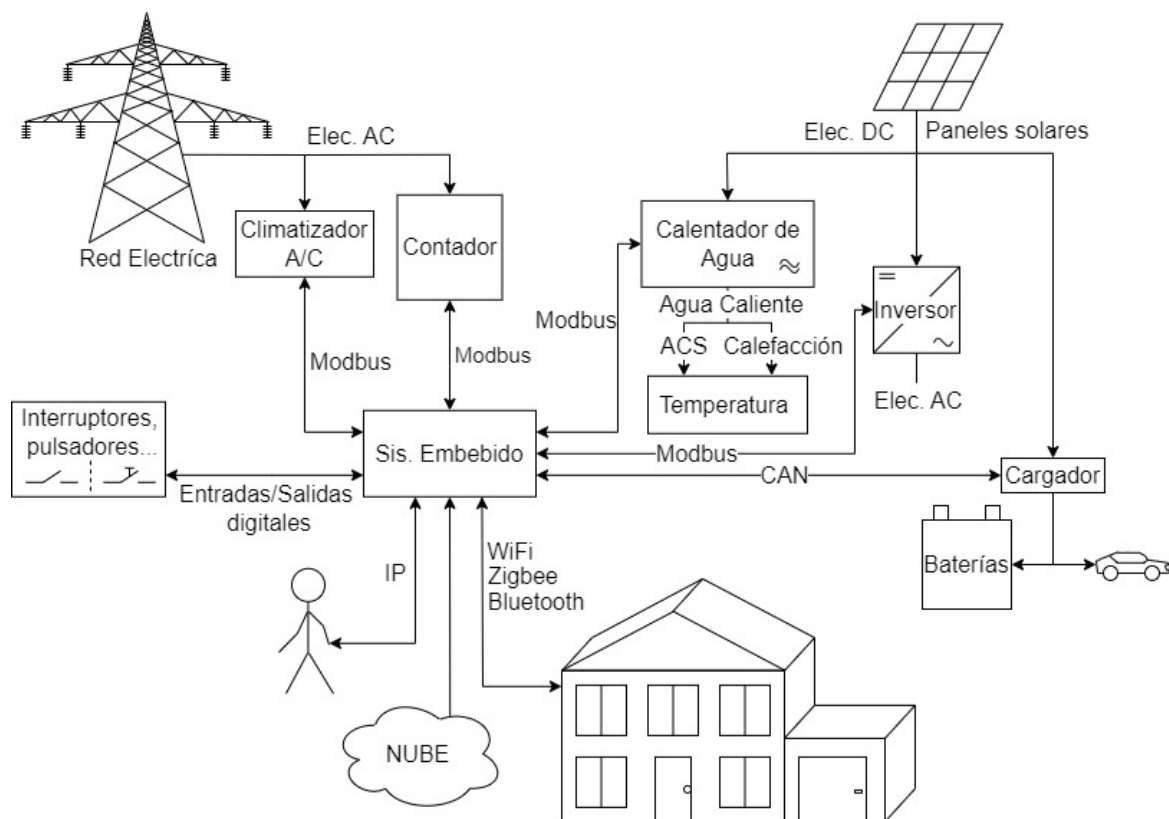


Figura 2.3: Entorno a controlar

2.3. Protocolos de comunicación

Para controlar los distintos dispositivos que ocupan el espacio a gestionar, se necesitarán distintos protocolos de comunicación con los que poder obtener la información necesaria y actual del entorno, y con ella actuar de la forma debida.

2.3.1. Protocolos industriales

- **Modbus**

Modbus es un protocolo optimizado y ampliamente usado que garantiza la transmisión de datos ETHERNET ultrarrápida. Permite conectar un dispositivo maestro (p.ej., el controlador) y varios dispositivos esclavos (p.ej., inversores, calentadores...) mediante varias versiones, una para interfaz serie (RS-232 y RS-485) y otra para ETHERNET. Destacan Modbus TCP basada en un modelo cliente/servidor o Modbus RTU con transmisión asíncrona en serie[26].

- **CAN Bus**

CAN bus (*Controller Area Network*) es un protocolo de comunicación en el que distintos dispositivos pueden intercambiar información entre sí sin necesidad de un host. Se basa en mensajes y está diseñado específicamente para aplicaciones automotrices; no obstante, se aplica en otras áreas como aerospacial o automatización industrial. Permite que los componentes se comuniquen en un bus de datos de uno o dos cables y en el que la tensión que pasa por estos cables varía según si está inactivo o transmitiendo[27].

2.3.2. Protocolos domésticos

- **WiFi**

WiFi (*Wireless Fidelity*) es una conexión inalámbrica que permite el acceso a internet de los dispositivos (móviles, ordenadores, etc.) Para la conexión se requerirá de un router, que está conectado a internet por cable y distribuye la conexión al resto de dispositivos inalámbricamente[28].

- **Bluetooth**

Estándar de conectividad inalámbrica que permite la transmisión de datos entre dispositivos a corto alcance evitando presencia de cables o conectores con interacción sencilla y rápida. Actualmente hay versiones para IoT como la 5.0 y actualizaciones que intentan mejorar la conectividad[29].

- **Zigbee**

Protocolo para capas bajas de comunicación que destaca por su bajo consumo, baja tasa de transferencia de datos, tamaño reducido y bajo coste. Gestiona automáticamente la red lo que permite corregir situaciones automáticamente evitando acceder físicamente a los equipos. Es muy usado en domótica, pudiendo integrar módulos Zigbee de forma muy sencilla en distintos elementos comunes[30].

- **IP**

Siendo este un protocolo tanto industrial como doméstico, es uno de los pilares básicos de Internet, funcionando como *distribuidor* de paquetes de datos que se

transmiten por caminos sin recepción garantizada (esto se resuelve con TCP/IP). Se utiliza para direccionamiento habitualmente, identificando cada máquina[31].

2.3.3. Comunicación con la nube

Red mundial de servidores con funciones únicas conectados para funcionar en un único ecosistema. Estos servidores almacenan, administran datos o entregan servicios entre otras características para el uso de los usuarios. A partir de la nube se puede tomar información o almacenarla para poder utilizarla según las necesidades.[32]

2.4. Necesidades de Software

Con lo anteriormente mencionado, será necesario un sistema que sirva como controlador de distintos dispositivos. Se tratará de un Software que pueda comunicarse mediante protocolos industriales con distintos equipos del entorno, y al mismo tiempo, mediante domésticos para obtener información de sensores o activar/desactivar diferentes dispositivos. A su vez, debe poder gestionar información de los usuarios locales.

En el caso de no haber conexión online se deberá permitir el uso offline de distintos dispositivos; posibilidad de acceso remoto al software desde distintos puntos o un posible acceso a la nube para poder tomar distintos datos de interés.

Será adecuado que el software sea de *código abierto* (***Open Source***), presentando una interfaz web agradable y con uso sencillo. A su vez, será interesante poder realizar autodiagnósticos para conocer en todo momento si hay algún fallo o problema.

2.5. Necesidades de Hardware

A partir de lo analizado, será necesario un sistema hardware que permita realizar las distintas comunicaciones y posibilite conexiones a distintos pulsadores o interruptores para que el usuario decida lo que más le interesa en cada instante.

Se necesitará que el hardware disponga de una CPU principal en la que se instalará el Sistema Operativo y el software que controlará el entorno, permitiendo la posibilidad de realizar actividades con una interfaz de alto nivel. Deberá tener una memoria que sea suficiente para suplir las necesidades.

Finalmente, el sistema embebido debe ser seguro, cómodo y fácil de usar, aportando diferentes posibilidades a la hora de utilizarlo, para tener mayores facilidades al realizar las distintas conexiones, comunicaciones y ejecuciones del software.

2.6. Tablas comparativas

A partir de las necesidades, se realiza una comparativa entre distintos softwares y hardwares dedicados a IoT y automatización domótica, para así escoger los más idóneos para alcanzar los objetivos. ***Comparativa de Software:*** Tabla 2.1 y Tabla 2.2, ***Comparativa de Hardware:*** Tabla 2.3, Tabla 2.4 y Tabla 2.5.

Software	Protocolos de comunicación	Accesibilidad	Otras características	Sistema operativo y lenguaje
OpenHAB	-Industriales: Modbus, CAN bus, Ethernet -Domésticos: Bluetooth/BLE, Zigbee, MQTT (uso de broker como Mosquitto), WiFi	-Comunicación con la nube: OpenHAB Cloud o servidor propio -Modo offline -Acceso remoto: VPN, servicios nube o Reverse Proxy	-Interfaz de usuario: Interfaz web y aplicación móvil -Open Source: Abierto -Autodiagnóstico: Disponible	-Sistema operativo: Linux, macOS, Windows, Raspberry Pi, apps para web, iOS o Android. -Lenguaje: Blockly (gráfico), JavaScript. Opcionales: Python, Ruby, Groovy o Java.
Domoticz	-Industriales: Modbus, CAN bus, Ethernet con TCP/UDP, IP y HTTP -Domésticos: Bluetooth, Zigbee, WiFi, MQTT (uso de broker como Mosquitto)	-Comunicación con la nube: servicios nube disponibles -Modo offline -Acceso remoto: VPN, servicios nube o servidor web Proxy	-Interfaz de usuario: Interfaz web HTML5 escalable, para escritorio como para móviles -Open Source: Abierto -Autodiagnóstico: Disponible	-Sistema operativo: Linux, Windows, macOS o dispositivos embebidos. -Lenguaje: Blockly (gráfico), Python, LUA o dzVents.
Rex Control System	-Industriales: Modbus TCP, CAN bus, CANopen, Ethernet -Domésticos: Bluetooth, Zigbee, WiFi, MQTT	-Comunicación con la nube: administración de datos en la nube -Modo offline -Acceso remoto disponible	-Interfaz de usuario: Servidor web incorporado, para escritorio, tablet o teléfono inteligente basada en HTML5 -Open Source: Pago -Autodiagnóstico: Disponible	-Sistema operativo: Windows, macOS, Linux, iOS o Android. Para Raspberry Pi o PLCs también. -Lenguaje: lenguaje de programación basado en bloques de funciones (gráfico).
CODESYS	-Industriales: CANopen, CAN bus, Ethernet, PROFINET, Modbus -Domésticos: Bluetooth, Zigbee, WiFi, MQTT	-Comunicación con la nube: acceso a la nube posible no obligatorio -Modo tanto offline como en línea -Teleacceso a través de Ethernet o bus campo	-Interfaz de usuario: Interfaz única con gran variedad de add-ons -Open Source: Pago -Autodiagnóstico: Disponible	-Sistema operativo: Windows, macOS, Linux o sistemas embebidos como la Raspberry Pi. -Lenguaje: Structured Text (ST), Ladder Diagram (LD), Function Block Diagram (FBD) o Sequential Function Chart (SFC).
OpenRemote	-Industriales: Serial, TCP, Modbus, Can bus, Ethernet -Domésticos: Bluetooth, Zigbee, WiFi, HTTP, MQTT (broker)	-Comunicación con la nube: acceso disponible a la nube -Modo tanto offline como en línea -Acceso multiusuario y remoto disponible	-Interfaz de usuario: Interfaz web y aplicación móvil para monitoreo y mantenimiento -Open Source: Abierto -Autodiagnóstico: Disponible	-Sistema operativo: Linux, Windows 10 y posteriores versiones, también en SBCs como la Raspberry Pi. -Lenguaje: Programación mediante reglas When-Then, o mediante reglas de flujo de manera gráfica. También puede usarse Groovy.
Edgexfoundry	-Industriales: Modbus TCP, Ethernet, Can bus -Domésticos: Bluetooth/BLE, Zigbee, WiFi, MQTT (broker, p.ej. Mosquitto)	-Comunicación con la nube: funcionamiento posible tanto de manera local como en la nube -Modo offline -Acceso remoto mediante conexión cifrada con SSH	-Interfaz de usuario: Interfaz web intuitiva y amigable -Open Source: Abierto -Autodiagnóstico: Disponible	-Sistema operativo: Linux, Windows, macOS o en cualquier SBC como puede ser una Raspberry Pi. -Lenguaje: Programación en Go Lang y C (uno u otro según las necesidades).

Tabla 2.1: Comparativa de Softwares 1

Software	Protocolos de comunicación	Accesibilidad	Otras características	Sistema operativo y lenguaje
ROS	-Industriales: Modbus TCP, Ethernet, Can bus -Domésticos: Bluetooth/BLE, Zigbee, WiFi, MQTT (cliente comunicando mediante un broker)	-Comunicación con la nube: posibilidad de uso -Apps instalables de uso offline -Acceso remoto mediante TeamViewer o con línea de comandos (ssh).	-Interfaz de usuario: Interfaz web para almacenar y visualizar datos de forma intuitiva -Open Source: Abierto -Autodiagnóstico: Disponible	-Sistema operativo: Linux, Windows, macOS o en plataformas integradas como micro-ROS. -Lenguaje: Programación en Python, C++ y Lisp, también hay alguna biblioteca en Java y Lua.
Home Assistant	-Industriales: Modbus serial, TCP o UDP, CAN bus, Ethernet -Domésticos: Bluetooth/BLE, Zigbee, WiFi, MQTT (con broker propio o add-on como Mosquitto)	-Comunicación con la nube: uso y análisis de datos de la nube -Funcionamiento offline con datos locales -Acceso remoto mediante Remote UI accediendo a la nube	-Interfaz de usuario: Interfaz web completa, aplicaciones móviles adicionales -Open Source: Abierto -Autodiagnóstico: Disponible	-Sistema operativo: Home Assistant Yellow (propio), Raspberry PI, Windows, macOS, Linux o sistemas genéricos x86-64. -Lenguaje: Programación principalmente en YAML, sin embargo está escrito y permite su uso en Python.
Microsoft Azure IoT Edge	-Industriales: Ethernet, Modbus TCP o CAN bus -Domésticos: Bluetooth/BLE, Zigbee, WiFi, NFC, LoRaWAN, MQTT (uso de IoT SDKs par construir clientes de dispositivos IoT)	-Comunicación con la nube: análisis de los datos locales en la nube -Toma de decisiones fuera de línea (modo offline), datos locales -Administración remota de los dispositivos IoT Edge en la nube	-Interfaz de usuario: Interfaz web basada en la nube -Open Source: Pago -Autodiagnóstico: Disponible	-Sistema operativo: Linux, Windows y otros sistemas embebidos como Raspberry PI. -Lenguaje: Programación en Java, .NET Core 3.1, Node.js, C y Python.
Nodered	-Industriales: Modbus, CAN bus, Ethernet IP -Domésticos: Bluetooth/BLE, Zigbee, WiFi, MQTT (conexión con un broker ejecutado localmente como Mosquitto)	-Comunicación con la nube: eventos ejecutables en la nube (Amazon o Azure) -Eventos también ejecutables de manera local (offline) -Acceso remoto mediante VNC o mediante una VPN	-Interfaz de usuario: Interfaz web/navegador -Open Source: Abierto -Autodiagnóstico: Disponible	-Sistema operativo: Linux y versiones como Ubuntu o Debian, Raspberry PI, Fedora o CentOS, Windows o Android (experimental). -Lenguaje: Programación mediante diagramas de bloques unidos por nodos. Funciones escritas en JavaScript. Basado en Node.js.
Homebridge	-Industriales: Modbus, CAN bus, Ethernet IP -Domésticos: WiFi, BLE/Bluetooth, Zigbee, MQTT (plugin con una API MQTT)	-Comunicación con la nube: administración y análisis de datos en la nube -Modo offline accediendo de manera local -Acceso remoto con distintas maneras, como mediante ssh	-Interfaz de usuario: Interfaz web disponible y uso de dispositivos Apple -Open Source: Abierto -Autodiagnóstico: Disponible	-Sistema operativo: macOS, Windows 10, Linux, Docker, Raspberry PI y otras plataformas. -Lenguaje: Programación en TypeScript y JavaScript, entorno basado en Node.js.

Tabla 2.2: Comparativa de Softwares 2

Módulo	CPU	Memoria y almacenamiento	Conectividad	Otras características
gS01	-Nombre: Intel® Atom E39XX (XX=30, 40 ó 50) -Fabricante: Intel -Tipo: Intel Atom® Processor X Series -Nº de cores: 2/4 -Max CPU Clock: 2(1.8) -Real-Time Co-CPU: (Única CPU con 2 o 4 cores)	-RAM: hasta 8GB DDR4 -eMMC: hasta 64GB	-1 SD, 6 USB2.0, 2 USB3.0, 3 UART, 5 I2C, 1 SPI, 19 GPIO -CAN: 2 (Pero solo para la FPGA) -Ethernet: 1 Gigabit Ethernet -WiFi: - -Bluetooth: -	S.O.: Soporta plataformas Windows y Linux, Linux BSP rt (OSADL) or Android BSP. Dispone de FPGA de Intel Cyclone, HDA y I2S para tratamiento de audio, el módulo iría soldado directamente en la carrier.
Digi Connect-Core 93	-Nombre: NXP® i.MX 93 -Fabricante: NXP -Tipo: Cortex®-A55 (ARM) -Nº de cores: 2 -Max CPU Clock: 1.7 -Real-Time Co-CPU: Cortex-M33 a 250 MHz (ARM)	-RAM: hasta 2GB LPDDR4 -eMMC: hasta 32GB	-1 uSDHC, 2 USB2.0, 8 LPUART, 8 I2C, 2 I3C, 8 LPSPI, 112 GPIOs, 1 ADC, 4 PWM -CAN: 2 CAN/CAN-FD -Ethernet: 2x 10/100/1000 Mbps -WiFi: Wi-Fi 6 inalámbrico 802.11ax de doble banda 1x1 -Bluetooth: Bluetooth® 5.2	S.O.: Digi Embedded Yocto Linux. Tiene una NPU para aplicaciones AI/ML. Integra seguridad, identidad y privacidad del dispositivo Digi TrustFence. Dispone de 3 módulos de interfaz de audio síncrono (SAI).
VAR-SOM-MX93	-Nombre: NXP® i.MX93 -Fabricante: NXP -Tipo: Cortex-A55 (ARM) -Nº de cores: 2 -Max CPU Clock: 1.7 -Real-Time Co-CPU: 250MHz Cortex-M33 (ARM)	-RAM: 512MB-2GB LPDDR4 -eMMC: 8-128GB	-1 SD, USB2.0, 4 UART, 5 I2C, 3 SPI, 1 JTAG, 4 ADC -CAN: 2 -Ethernet: 2x 10/100/1000 Mbps -WiFi: Certified single-band 802.11 b/g/n or dual-band 802.11 ac/a/b/g/n -Bluetooth: BT/BLE 5.2	S.O.: Linux (Yocto, Debian), FreeRTOS. Pertenece a la familia Pin2pin de variscite. RTC en la carrier, aceleración NPU para AI/ML. Hay driver para auriculares, micro digital o analógico y distintas interfaces serie para audio digital.
VAR-SOM-MX8	-Nombre: NXP® i.MX 8QuadMax / 8QuadPlus -Fabricante: NXP -Tipo: 2x Cortex™-A72 + 4x Cortex™-A53 (ARM) -Nº de cores: 5 a 6 -Max CPU Clock: 1.6 GHz Cortex™-A72, 1.2 GHz Cortex™-A53 -Real-Time Co-CPU: 2x 266MHz Cortex™-M4F (ARM)	-RAM: 2-8GB LPDDR4 -eMMC: 8-128GB	-1 SD, USB3.0/2.0, 5 UART, 4 I2C, 4 SPI, 1 JTAG, 1 keypad (5x4 matrix), PCI-Express 3.0 -CAN: 2 CAN/CAN-FD (Flexible Data-Rate) -Ethernet: 2x 10/100/1000 Mbps -WiFi: Certified 802.11 ac/a/b/g/n -Bluetooth: BT/BLE 5.2	S.O.: Linux (Yocto, Debian), FreeRTOS, Android. Pertenece a la familia Pin2Pin de variscite. Procesador HiFi4 DSP para procesamiento de señales digitales. Hay 2 GPU Vivante GC7000VX, driver para auriculares, microfono digital o analógico e interfaz serie para audio Digital de tipo I2S(SAI). RTC en carrier.
Aikri-X10-6S	-Nombre: Qualcomm® QCS610 -Fabricante: Qualcomm -Tipo: Qualcomm® Kryo™ 460 (basados en ARM) -Nº de cores: 8 (2 de alto rendimiento, 6 de bajo consumo) -Max CPU Clock: 2.2/1.8 -Real-Time Co-CPU: -	-RAM: 4GB LPDDR4 -eMMC: 16GB	-1 SD, 1 USB3.1, 2 USB2.0, 1 UART, 3 I2C, 2 SPI, GPIOs -CAN: - -Ethernet: 1 Gigabit Ethernet -WiFi: Qualcomm® Wi-Fi (WCN3980) 802.11a/b/g/n/ac 2.4/5.0 GHz -Bluetooth: BT 5.0	S.O.: Linux. Disponibilidad de comunicación vía GPS. Sensores: acelerómetro y giroscopo, magnetómetro, proximidad, temperatura y presión. Controlador gráfico Adreno 608, para audio un codificador Qualcomm WCD9341 y 3 entradas analógicas y digitales de micro.

Tabla 2.3: Comparativa Hardware (System on Modules)

SBC	CPU	Memoria y almacenamiento	Conectividad	Otras características
Digi Connect-Core 8X SBC Pro	-Nombre: NXP i.MX8QuadXPlus -Fabricante: NXP -Tipo: Cortex-A35 -Nº de cores: 4 -Max CPU Clock: 1.2 -Real-Time Co-CPU: Cortex-M4F a 266 MHz	-RAM: 2GB LPDDR4 -eMMC: 16GB	-1 MicroSD, 1 USB2.0, 1 USB3.0 en tipo-C, 2 UART, I2C, SPI, GPIOs, 1 JTAG -CAN: 2 -Ethernet: 2x Gigabit Ethernet 10/100/1000M -WiFi: 2 x 2 MIMO 802.11a/b/g/n/ac Seguridad: WEP, WPA-PSK/WPA2-Personal -Bluetooth: BT 5.0	S.O.: Compatible con Yocto Project® Linux® y Android™ Seguridad mediante Digi TrustFence. Audio: 2 entradas, 1 salida de audio, 1 entrada de micrófono y 1 conector jack de 3.5mm.
EIC-Q660-500	-Nombre: Qualcomm Snapdragon™ 660 -Fabricante: Qualcomm -Tipo: Qualcomm® Kryo™ 260 (basados en ARM) -Nº de cores: 8 -Max CPU Clock: 2.2 -Real-Time Co-CPU: -	-RAM: 6GB LPDDR4 -eMMC: 64GB	-1 MicroSD, 2 USB2.0, 1 USB3.1 en tipo-C, UART, 2 I2C, 2 SPI, 16 GPIOs, 1 JTAG -CAN: - -Ethernet: 1x 10/100 Mbps (Via USB2.0) -WiFi: IEEE 802.11 a/b/g/n/ac -Bluetooth: BT/BLE 5.1	S.O.: Android 8.1 Sensores: acelerómetro y giroscopo, magnetómetro, proximidad. Comunicación vía GPS con un Qualcomm GPS integrado, GPU Adreno 512 para multimedia y para audio hay 2 MIC analógicos, 1 altavoz y 1 Jack de 3.5mm con MIC, entre otros.

Tabla 2.4: Comparativa Hardware (Single Board Computers)

Referencia del kit de desarrollo / SBC	Módulos compatibles y CPU	Multimedia	Interfaces y expansiones	Conectividad
Digi ConnectCore Kit de desarrollo 93	-Módulos compatibles: Digi ConnectCore® 93 -Nombre CPU: NXP® i.MX 93 -Fabricante CPU: NXP	-HDMI: - -Audio: 3x Synchronous Audio Interface (SAI) modules (up to 4 lanes) supporting I2S, AC97, TDM, codec/DSP and DSD interfaces -Interfaz cámara: 1x 1080p 60 MIPI-CSI (2-lane, 1.5 Gbps/lane)	-USB: 2x USB 2.0 OTG -High-Speed Expansions: SD: 1 interfaz uSDHC -Puertos Serie: -	-x8 UART, x8 I2C, x8 SPI, x4 PWM, 12 bit ADC (hasta 4 canales), hasta 112 GPIOs -CAN: 2 CAN/CAN-FD (Flexible Data-Rate) -Ethernet: 2x 10/100/1000M -WiFi: Wi-Fi 6 802.11ax dual-band 1x1 wireless -Bluetooth: BT 5.2
Symphony-Board	-Módulos compatibles: Todos los de las familias VAR-SOM-MX8 y VAR-SOM-MX9 -Nombre CPU: según módulo CPU NXP® i.MX de series 8 o 9. -Fabricante CPU: NXP	-HDMI: 2.0a - Necesita de la extensión VAR-EXT-HDMI -Audio: Headphone: conector 3.5mm, Line-in: conector 3.5mm, Digital Microphone: on-board -Interfaz cámara: Serial camera: Interfaz no incluida necesita la extensión (VCAM-5640S), Parallel camera: necesita de "camera extension board"	-USB: Host: 1 port, Type A connector, USB2.0; OTG: 1 port, Type C connector, USB3.0/2.0 TypeC -High-Speed Expansions: PCIe: conector mini-PCIe; SATA: SATA Connector; Tarjeta SD: SD Card Socket -Puertos Serie: Debug port: USB-SERIAL bridge, Micro USB type AB	-UART, I2C, SPI, JTAG (header en SoM), GPIOs, Timer, QSPI, ADC -CAN: Disponible -Ethernet: 2x 10/100/1000 Mbps, RJ45 -WiFi: On SoM -Bluetooth: On SoM -Display: HDMI, LVDS y DSI -RTC
Aikri-X10-6D	-Módulos compatibles: Aikri-X10-6S -Nombre CPU: Qualcomm® QCS610 -Fabricante CPU: Qualcomm GPU para multimedia Adreno 608	-HDMI: - -Audio: Qualcomm® WCD9341 audio codec on-board, 1x Headset / Lineout, 3x Analog MIC input, 3x Digital MIC input -Interfaz cámara: 3x 4-Lane MIPI CSI	-USB: 1x USB 3.1, 2x USB 2.0 TypeC -High-Speed Expansions: SD: 1x SDC 3.0 -Puertos Serie: -	-x1 UART, x3 I2C, x2 SPI, x2 I2S, GPIOs(disponible en conector board-to-board) -CAN: - -Ethernet: 1x Gigabit Ethernet -WiFi: Qualcomm® Wi-Fi (WCN3980) 802.11a/b/g/n/ac - 2.4/5.0 GHz -Bluetooth: BT5.0

Tabla 2.5: Comparativa Hardware (Otras SBCs y Kits de desarrollo)

2.7. Conclusiones y decisiones

Como se ha visto, se necesita un sistema basado en un software que controle un espacio inteligente y las energías que accedan a este, para que el consumo sea más eficiente y los usuarios dispongan de mayor bienestar. A su vez, será necesario un hardware que funcione como núcleo de este entorno, con buena conectividad y capaz de gestionar las distintas exigencias que ocurran en cada instante.

En base a las tablas previas, para decidir que software elegir primero analiza si son *Open Source*, descartando por ello *Rex Control System*, *Codesys* y *Microsoft Azure IoT Edge*. Se descarta *ROS* ya que se encuentra más dedicado a programación de robots que para otros ámbitos. *Homebridge* por su parte, se encuentra limitado al estar enfocado a dispositivos Apple. Se descartarán por su lenguaje de programación o falta de experiencia *Nodered*, *Domoticz* y *OpenRemote* mediante bloques, y *OpenHAB* o *EdgeXFoundry* con lenguajes de programación principales son JavaScript y Go Lang respectivamente, en los que no se dispone de experiencia previa.

Por tanto, se escoge ***Home Assistant***, software ideal que permite realizar con facilidad los distintos protocolos de comunicación, accesibilidad a la nube, trabajar de manera offline y remota adecuadamente y con una interfaz web para su uso muy intuitiva e interesante. Su programación será mediante Python, lenguaje en el que se tiene experiencia. Dispone de una amplia comunidad y está muy bien valorado en el sector de la automatización domótica.

Respecto al hardware, se decide escoger un SOM con su kit de desarrollo, ya que disponen de mejores características que las versiones basadas en única SBC.

Teniendo claro lo anterior, se comienza eligiendo el procesador que más interesa. El SOM gs01 con procesador Intel será descartado ya que no permite WiFi o Bluetooth, y CAN solo será posible para FPGA. Por otro lado, el SOM con procesador Qualcomm no permite comunicación CAN Bus. Se llega entonces a la conclusión de que los procesadores más adecuados serán los de NXP, entre los que se encuentran los SOM de Digi y Variscite principalmente, con muy buenas características para la aplicación en la que se va a utilizar. Se escogerá un módulo de *Variscite*, ya que estos tienen ciertas ventajas al pertenecer a la *Pin2pin Family*, con ventajas en cuanto a escalabilidad, calidad/precio, facilidad de desarrollo, integración y tiempo de vida. Su SBC será la ***Symphony Board***.

Se decide utilizar el módulo ***VAR-SOM-MX93***, con buenas características en cuanto a CPU, disponiendo de un co-procesador en tiempo real que podría servir de manera opcional en algunos casos para ejecutar órdenes en tiempo real, buena memoria y almacenamiento, amplia conectividad, uso de linux y en el caso de que se necesitase dispone de una NPU para AI/ML. Se escoge sobre el módulo *VAR-SOM-MX8* ya que aunque este es de una gama más alta, no será necesario disponer de mayores prestaciones lo que aumentará el coste del módulo.

Capítulo 3

Hardware

3.1. Introducción

En este capítulo se va a explicar y analizar el hardware a utilizar, comenzando por el estudio de sus características y funcionamiento. Posteriormente, se desarrollará y trabajará sobre él explicando el procedimiento realizado.

Como se ha mostrado en el capítulo 2, se ha realizado un análisis entre diferentes hardwares con diferentes tipos de procesadores, llegando a la conclusión y elección de un SoM (System on Module) con su kit de desarrollo del fabricante Variscite, con procesador NXP y una co-CPU para efectuar tareas en tiempo real. El módulo elegido finalmente por sus adecuadas características es el ***VAR-SOM-MX93***.

Para la realización de este capítulo se tomará principalmente la información de la web oficial y de la wiki del propio SoM de Variscite[33].

3.2. Arquitectura del Hardware

3.2.1. System on Module

En primer lugar se va a tratar el dispositivo principal y que funcionará como núcleo de todo el hardware a utilizar, será el módulo ***VAR-SOM-MX93*** del fabricante Variscite y será el que contenga al procesador y microcontrolador en tiempo real sobre los que se ejecute el software y manden las distintas indicaciones a realizar. Este módulo se colocará sobre la SBC Symphony Board.

Este SoM es un módulo de bajo consumo que integra una AI/ML NPU (Unidad de Procesamiento Neuronal) para aceleración de sistemas embebidos en un precio atractivo. A su vez, ofrece una arquitectura de energía flexible, características de seguridad integradas y un amplio conjunto de periféricos y opciones de conectividad. Se trata de un SoM compatible con la familia VAR-SOM Pin2Pin que proporciona máxima escalabilidad, buena relación calidad/precio y facilidad de, integración.

En cuanto a sus especificaciones, el módulo VAR-SOM-MX93 dispone de una CPU ***NXP i.MX93*** de arquitectura ARM *Cortex-A55*. Se forma de 2 núcleos y su reloj máximo de la CPU llega hasta los 1.7 GHz. Por otro lado, incluye un *co-procesador* en tiempo real también de ARM *Cortex-M33* que llega hasta 250 MHz.

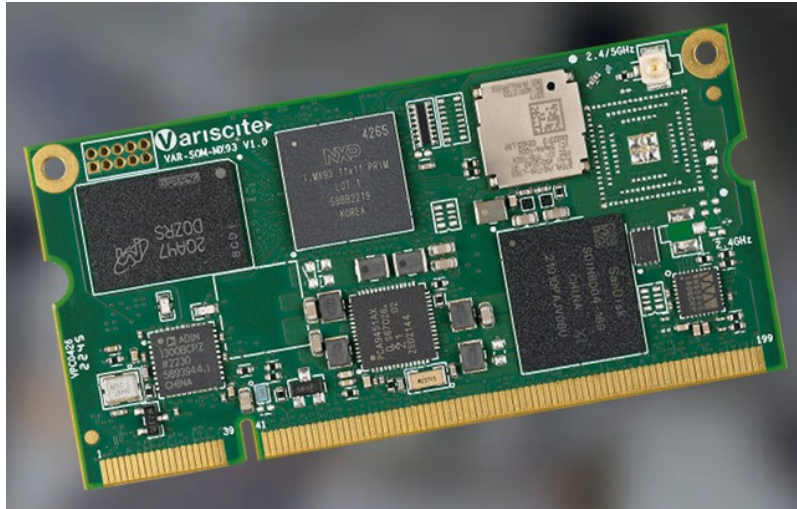


Figura 3.1: VAR-SOM-MX93 (Fuente: Variscite)

Cabe dar importancia al **procesador de tiempo real** siendo una ventaja frente a otras SBCs típicas como es una Raspberry PI. El uso de un procesador en tiempo real permite dar libertad al procesador principal en distintos aspectos, y facilitando que se puedan realizar ciertas lecturas de distintas entradas y con ellas actuar sobre respectivas salidas de manera muy rápida trabajando en tiempo real. A su vez, forma una arquitectura dual con el procesador principal que da gran flexibilidad y aporta ventajas frente al uso de versiones como la mencionada Raspberry PI.

Respecto a su memoria y almacenamiento, cuenta con una *RAM* de 512 MB a 2 GB, a su vez hay una *eMMC*, capacidades más que suficientes para el objetivo a lograr.

Este módulo posee de distintas características multimedia como son aceleración de gráficos 2D/3D mediante un motor de aceleración, a la vez que tiene una interfaz de cámara. Como ya se había nombrado previamente, se trata de un SoM con *AI/ML NPU*. Por otra parte, dispone de características para display encontrando un RGB paralelo, *MIPI-DSI*, *LVDS* y controlador táctil tanto resistivo como capacitivo. También cabe nombrar su capacidad de audio, teniendo accesibilidad a un driver de auriculares, micrófono digital o analógico (estéreo), una interfaz serie de audio digital y dispone de línea de entrada y salida de audio.

Por otro lado, posee distintos tipos de conectividades entre las que se encuentran *SD/MMC*, x2 Host/Dispositivo *USB2.0*, *UART*, *I2C*, *SPI*, *CAN Bus*, *JTAG* y *ADC*. Habrá *RTC* en la carrier. Respecto a la conexión a la red, se encuentra Ethernet, WiFi certificado de banda simple o doble y finalmente bluetooth *BT/BLE5.2*.

Para su uso se apoya de un sistema operativo *Linux* que puede ser *Yocto*, *Debian* o *Boot2Qt*, y por otra parte *FreeRTOS* para el co-procesador en tiempo real. Tiene unas dimensiones de 67.8 x 33mm y soporta unas temperaturas de -40 a 85°C.

3.2.2. Single Board Computer

La *Symphony Board* es una Single Board Computer (SBC) que sirve como plataforma de desarrollo y producto final. En conjunto con el módulo VAR-SOM-MX93 u otros de la familia ofrece un amplio abanico de conectividades e interfaces.[34]

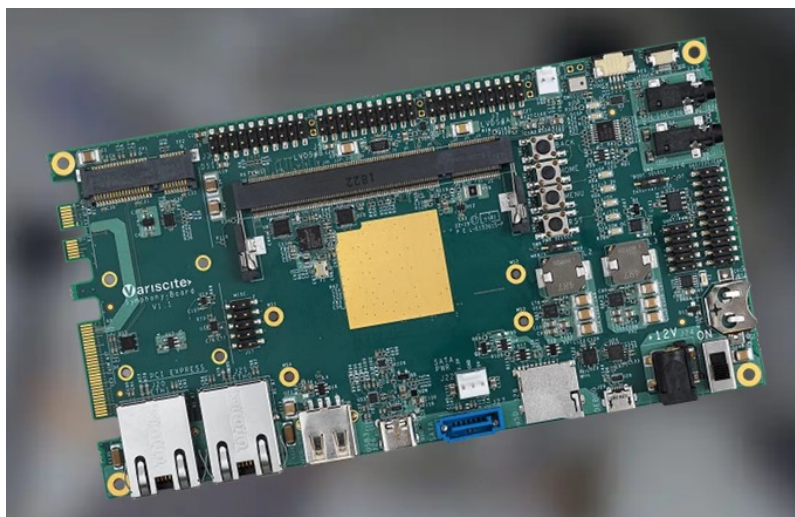


Figura 3.2: Symphony Board (Fuente: Variscite)

Esta SBC soporta sobre ella distintos módulos de Variscite entre los que se encuentran de forma más relevante el SoM VAR-SOM-MX93 y los de la serie VAR-SOM-MX8 (con procesadores de serie i.MX8).

Respecto a sus características multimedia, la Symphony Board dispone de dos Headers para *LVDS*, *HDMI 2.0* (con extensión) y un header *DSI* entre otros. En cuanto a audio, posee de un conector de 3.5mm para auriculares y un micrófono digital *on-board*. Dispone de conexión a panel táctil tanto resistivo de 4 cables, como capacitivo. Para interfaz cámara necesita de una extensión tanto para serie como para paralela.

En cuanto a sus interfaces, se encuentran un puerto Host USB 2.0 tipo A, y un puerto USB tipo C. Por otro lado, posee de expansiones de alta velocidad como *PCI-Express* con un conector *mini-PCIe*, *SATA* con un conector SATA y una tarjeta SD a través de un Socket para tarjetas SD. Adicionalmente, se posibilita un puerto de Debug con puerto serie Micro USB, y también de otras distintas expansiones como UART, CAN Bus, SPI, I2C, GPIOs, JTAG, un Timer y ADC, entre otras.

Finalmente cabe destacar que dispone de 2 conectores Ethernet. La SBC tiene unas dimensiones de 16.99 cm x 8.96 cm x 1.6 cm, se alimenta a 12V DC con un puerto jack y dispone de un socket RTC para batería de respaldo.

3.2.3. Funcionamiento y uso

Para que la placa pueda funcionar se deben tener en cuenta ciertas claves de manera previa. El procesador NXP que se encuentra en este SoM VAR-SOM-MX93 necesita de un sistema operativo, y su instalación será posteriormente el primer paso que se realice tras iniciar el módulo. El Sistema Operativo que se utilizará en este módulo será Linux, y se encuentran disponibles distintas versiones de Yocto y Debian. Variscite recomienda el uso de Yocto que es un sistema operativo Linux dedicado a aplicaciones en sistemas embebidos, y permitiendo gran variedad de posibilidades[35].

Con el sistema operativo instalado se podrá proceder a la instalación del software y realizar el control como se verá en el siguiente capítulo.

3.3. Diseño del Hardware

En esta sección se va a diseñar y explicar el hardware utilizado, a su vez se argumentarán las tomas de decisiones y su procedimiento. Como se ha visto en apartados anteriores, se va a utilizar el kit de desarrollo de *Variscite* basado en el System on Module *VAR-SOM-MX93*.

3.3.1. Comenzando con el módulo

El kit de desarrollo del módulo VAR-SOM-MX93 se compone un display LCD táctil y de la ya mencionada Symphony Board. A su vez, se incluye ya montado sobre la SBC, el módulo VAR-SOM-MX93. A pesar de que no se va a sobrecargar al procesador, se dispone de un disipador de calor para que en caso de un aumento elevado de su temperatura, esta pueda ser disipada de manera adecuada.

En primer lugar, y siguiendo la guía *Quick Start Guide* (disponible en la propia web de Variscite), se conectan el display LCD y la SBC. Posteriormente, se realiza *debug* desde un PC para inicializarlo ejecutando un Host. Por último, se alimenta la Symphony Board con ello el hardware ya estaría en condiciones para ser encendido.

Antes de activar el switch que permite alimentar la placa, se ejecutará un *Host* desde el PC como se acaba de mencionar. En el caso de utilizar un PC con Sistema Operativo Windows se puede instalar *Putty*, que se trata de un SSH de código abierto con el que se podrá acceder al procesador.

Se puede configurar mediante comunicación serie, es decir, accediendo desde el puerto USB que conecta al PC y SBC, o si fuera necesario mediante SSH introduciendo la dirección IP asignada al módulo. Se decide utilizar la primera opción de manera principal, ya que en una primera instancia no se tiene conocimiento de esa IP asignada, y en el caso de no tener conectado Ethernet a la SBC acceder con el puerto serie será posible, no siéndolo en el otro caso.

El entorno de Variscite dispone de dos métodos de arranque, desde la memoria flash eMMC, o desde una tarjeta microSD. El kit incluye una *microSD* pre-configurada con un Sistema Operativo Yocto, por lo que se comenzará iniciando desde esta tarjeta, colocando el switch de modo de arranque en la posición *SD*, se arrancará desde ella. Por último, se activa el interruptor de arranque.

Por otro lado, si se utilizara un Sistema Operativo base distinto a Windows, por ejemplo en Ubuntu, podría utilizarse un terminal host picocom o minicom. Para ello, se debe crear una máquina virtual de Ubuntu, mediante un software de generación de máquinas virtuales como puede ser *Virtual-Box*. La instalación de picocom, por ejemplo, se realizará mediante las siguientes instrucciones:

```
$ sudo apt-get install picocom  
  
$ sudo picocom -b 115200 /dev/ttyUSBX
```

Primero se instala picocom, utilizando *sudo* que dará permisos y acceso de administrador. Finalmente se lanza picocom mediante este último comando, sustituyendo la X por el número apropiado al puerto USB en el que se realice la conexión con la SBC.

Debido a que el Sistema Operativo base del PC a utilizar es Windows, se decide por el uso del host Putty, ya que su acceso será más rápido. No obstante, posteriormente para el uso del procesador de tiempo real sí que será necesario el acceso mediante Ubuntu, con el objetivo de ejecutar los comandos necesarios para su uso.

3.3.2. Yocto OS: tests y uso de la interfaz

Al arrancar inicialmente el módulo desde el host Putty en modo SD, se muestran en la terminal los mensajes de comienzo de Yocto y tras ello se accederá con el usuario **root** que permitirá el uso de permisos como administrador.

Para instalar el Sistema Operativo Yocto en el almacenamiento interno, tan solo será necesario escribir la siguiente línea en la terminal y seguir las instrucciones que se pidan por pantalla:

```
# install_yocto.sh
```

Después de esto, se podrá reiniciar el módulo tras cambiar el modo de arranque en la SBC a *INT* (modo interno). Ya desde el almacenamiento interno, se puede observar la correcta instalación de Yocto.

Variscite ofrece la posibilidad de crear una imagen propia de Yocto siguiendo los pasos que se pueden encontrar en su propia Wiki. Sin embargo, esta opción requiere del uso de la máquina virtual Ubuntu, en la que se instala la imagen. Este proceso necesita de una gran cantidad de recursos que esta máquina virtual no es capaz de suministrar. Tras intentar crear una imagen Yocto sin resultados positivos, se escoge la opción más óptima y rápida, siendo el uso de la imagen ya prediseñada de la microSD.

Para comenzar con el hardware, se realizan distintos tests de funcionamiento y uso de la interfaz que se pueden encontrar en la wiki de Variscite[35]. Con ellos se podrán probar distintas características del módulo y de la interfaz.

- *CPU number of cores:*
Este SOM dispone de dos cores, estando el primero siempre habilitado, se podrá actuar por tanto sobre el segundo. Este test realizado permite conocer que núcleos se encuentran activos y si se deseara se puede activar o desactivar alguno de ellos.
- *CPU temperature:*
El SOC i.mx93 dispone de un sensor de temperatura interno que puede ser leído y que se muestra en **miligrados Celsius**. La realización de este test o posible uso futuro es interesante ya que puede mostrar si el procesador se encuentra saturado sobrecalentándose pudiendo actuar rápidamente para evitar ese problema. A su vez, indicará según la temperatura si es necesario el uso de un disipador para solventar su aumento.
- *Tests de actuación de Ethernet:*
Para la prueba de la conexión Ethernet, en primer lugar, se comprobará que IP es la asignada al módulo tras establecer una conexión vía Ethernet. La SBC dispone de dos conexiones Ethernet por lo que al utilizar este test se podrá ver cual es la utilizada y que dirección se le ha asignado. Por otro lado, conociendo el puerto Ethernet utilizado y la IP, se podrán realizar tests de funcionamiento

como pueden ser TCP o UDP accediendo con el SOM como servidor y con otra terminal como cliente.

- *Wireless / WiFi:*

Con este test se permite conocer las distintas redes WiFi cercanas al SOM y permitir realizar una conexión con la deseada. Para ello se activará la radio WiFi del módulo para poder establecer conexión y se comprobarán las redes próximas a este. Finalmente se realizará una conexión a la red deseada de la siguiente forma:

```
# nmcli dev wifi connect <SSID> password <PASSWORD> ifname wlan0
```

Finalmente, se puede conocer el estado de la conexión ejecutando los comandos adecuados a lo que se quiere comprobar. En la siguiente figura se usa una instrucción que permite conocer las conexiones en general disponibles (pudiendo ver también la de Ethernet):

```
# nmcli con show
```

```
root@imx93-var-som:~# nmcli con show
NAME                                UUID                                TYPE      DEVICE
Supervisor eth0                    9a0191f1-2c78-362a-9170-8792cac495a6 ethernet  eth0
lo                                  2c2a9518-24df-4598-9e09-4126c491df68 loopback  lo
MOVISTAR_PLUS_5B76                 e725eb34-c396-4f39-884c-1916d64e88f4 wifi      wlan0
Wired connection 1                  59002dc8-0f07-335d-9047-9e993d2f5774 ethernet  --
```

Figura 3.3: Comunicación con WiFi

Se dispone de mayor capacidad de configuración y personalización, si se deseara se podrían realizar otros tests para poder comprobar otros temas interesantes en el módulo como pueden ser la comunicación Bluetooth, visualización de GPIOs o CAN bus, no obstante, no se ha profundizado más en ello.

Yocto es un Sistema Operativo muy interesante en relación con la configuración y trabajo en sistemas embebidos, sin embargo su forma de trabajar con paquetes y configurar distintos archivos es distinta a la habitual en otras versiones como Ubuntu o Debian. En Yocto se utilizan recetas y para ejecutar archivos se utiliza el comando “bitbake”. Por otra parte, a la hora de instalar Home Assistant, Yocto muestra algunos problemas y errores con la instalación de ciertos paquetes necesarios. Por tanto, debido a que Variscite ofrece otras versiones como es Debian, se decide por instalar una nueva versión de este Sistema Operativo sobre el módulo.

3.3.3. Instalación de Debian

Para la instalación de Debian se sigue el mismo procedimiento que se había realizado previamente con Yocto. Se descarga una imagen de Debian en una tarjeta SD y con ella se graba el Sistema Operativo en el almacenamiento interno del SOM.

Para descargar esta imagen de Debian, se accede al servidor de Variscite FTP[36], entrando con las credenciales aportadas por Variscite[35]. Dentro de este server se puede obtener información y versiones interesantes para la instalación de Sistemas Operativos sobre el módulo.

Finalmente se necesita un software que permita instalar la imagen en una tarjeta SD. Se utiliza Win32 Disk Imager que permite tal objetivo, y con el que se graba la

imagen descomprimida sobre una tarjeta SD.

Tras la instalación, se arranca desde la SD de nuevo. Una vez el arranque se complete, se instalará Debian en el almacenamiento interno del SOM:

```
# install_debian.sh
```

Por último, se puede volver a arrancar cambiando el modo a *INT*, ahora mediante la eMMC, y realizar los mismos tests u otros, como ocurría en Yocto o instalar el software de automatización Home Assistant.

3.3.4. Sensores

Es necesario el uso de sensores que permitan obtener información del entorno en el que se encuentran. Por ello, se van a utilizar sensores Zigbee que permitirán, mediante comunicación inalámbrica, obtener los datos de ese lugar en todo momento.

3.3.4.1. Zigbee 3.0 USB Dongle Plus

Dongle Plus es un adaptador USB Zigbee que sirve como punto de enlace entre dispositivos Zigbee en distintas plataformas de código abierto para poder controlarlos localmente sin necesidad de invertir en concentradores de distintas marcas, siendo una puerta de enlace universal.[37]

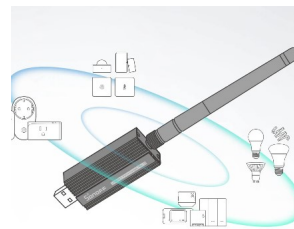


Figura 3.4: Dongle Plus (Fuente: SONOFF)

3.3.4.2. SONOFF S26R2ZB Zigbee Smart Plug



Figura 3.5: Smart Plug (Fuente: SONOFF)

Interruptor Zigbee controlable a distancia al ser conectado a cualquier enchufe. Puede servir como extensor del rango de trabajo ya que permite transmitir la señal entre dispositivos pudiendo tener menos preocupación respecto a los problemas de desconexión por larga distancia.[38]

3.3.4.3. SONOFF SNZB-01 ZigBee Wireless Switch

Pulsador Zigbee que permite mediante tres formas de activación realizar diferentes acciones. El pulsador permitirá realizar una cosa u otra dependiendo de si es pulsado mediante un pulso simple, mediante un doble pulso o mediante un pulso prolongado.[39]



Figura 3.6: SNZB-01 (Fuente: Amazon)

3.3.4.4. SONOFF SNZB-04 ZigBee Wireless Door/Window Sensor

Sensor Zigbee con dos partes, una contiene el circuito que comunica la señal de apertura o cierre, y otra con un pequeño imán que cuando esté lo suficientemente cerca de la otra, por efecto magnético, permitirá indicar el estado del sensor.[40]

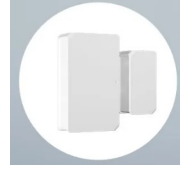


Figura 3.7: SNZB-04 (Fuente: SONOFF)

3.3.4.5. Aqara Sensor de movimiento RTCGQ01LM



Figura 3.8: Motion (Fuente: Xiaomipedia)

Sensor Zigbee que comunica la detección o falta de movimiento y la ocupación de una cierta estancia si se ha detectado movimiento recientemente, o por contrario, la no ocupación de la misma por falta de movimiento prolongada.[41]

3.3.4.6. SONOFF Zigbee LCD Smart Temperature Humidity Sensor

Sensor de Temperatura y Humedad de SONOFF que permite visualizar los datos mediante un display LCD. A su vez, vía Zigbee permite transmitir los valores tomados, pudiendo de esta forma almacenarlos para así conocer su evolución con el tiempo.[42]



Figura 3.9: Sensor T^a (Fuente: SONOFF)

3.3.5. Módulo de entradas/salidas digitales

Por último, se utilizará un módulo de **Dingtian** con 16 relés y distintas entradas digitales, controlado mediante un *ESP32*. Este módulo permitirá el control de los relés mediante distintos tipos de comunicación y su uso softwares de automatización como Home Assistant. Permite el acceso a una interfaz web en la que se podrá parametrizar o configurar el módulo para poderlo controlar las entradas y salidas. [43]

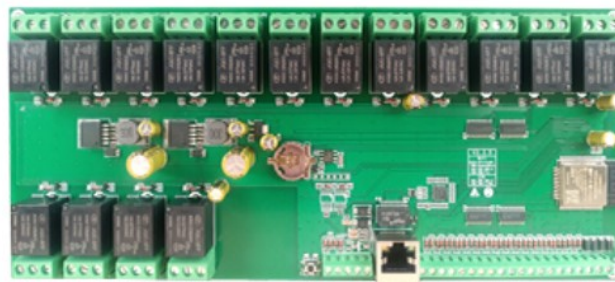


Figura 3.10: 16 Channel Relay Board (Fuente: dingtian-tech)

3.4. Co-Procesador en tiempo real Cortex-M33

En esta sección se va a analizar y utilizar el co-procesador en tiempo real de NXP Cortex-M33, que forma parte de la arquitectura dual de la CPU i.MX93 y que de manera equivalente al módulo Dingtian anterior puede emplear distintas entradas y salidas en tiempo real si fuese necesario.

El procesador Cortex-M33 es muy bueno en el ámbito IoT y aplicaciones integradas pudiendose aplicar en temas de seguridad pero sobretodo y en control de señales digitales. Logra una combinación óptima entre determinismo en tiempo real, eficiencia energética y productividad del software lo que abre la puerta a muchas aplicaciones y oportunidades. [44]

3.4.1. Puesta en marcha del co-procesador

El VAR-SOM-MX93 puede trabajar en tiempo real, y esto es gracias a este co-procesador, que puede ser muy útil para distintas aplicaciones que requieren un control en tiempo real como la activación de una salida al detectar un cambio en una entrada digital como puede ser un pulsador o un interruptor.

Para comenzar a utilizarlo, será necesario poder tener uso de un entorno con el que poder programarlo, construir posteriormente el proyecto realizado y finalmente ejecutarlo. Es por ello que Variscite ofrece el uso de una versión de **FreeRTOS**, que se trata de un sistema operativo para tiempo real. La versión será un SDK de **MCUXpresso**, que se encuentra diseñado para su uso en dispositivos NXP.

A partir de la wiki de Variscite [35] se consigue el procedimiento de instalación y uso de distintas demostraciones que sirven para probar el co-procesador con MCUXpresso.

En primer lugar será necesario usar un entorno base Linux, por lo que como se ha mencionado previamente, se utilizará una máquina Virtual Ubuntu. Como esta ya ha sido configurada, se podrá proceder a instalar directamente los paquetes requeridos.

Para comenzar, en una terminal se instalan paquetes necesarios y tras ello se descarga y finalmente instala la cadena de herramientas básicas GNU-ARM en un nuevo directorio:

```
$ tar xvf gcc-arm-none-eabi-10.3-2021.10-x86_64-linux.tar.bz2
```

Para terminar el proceso de instalación, se descarga el MCUXpresso SDK para el SOM:

```
$ git clone https://github.com/varigit/freertos-variscite -b mcuxpresso_sdk_2.13.x-var02
```

3.4.2. Aplicación del co-procesador en tiempo real

Una vez instalado, se pueden encontrar distintas demos a probar. De entre ellas se prueba una que permite parpadear un led ya que permite el uso de GPIOs.

Para comenzar a usar esta demo, esta se encuentra configurada totalmente y no hay que programar nada. Por lo tanto, se comienza construyendo el proyecto que luego se

ejecutará en el co-procesador. Variscite ofrece distintas formas de construir un proyecto, sin embargo se decide por la opción manual accediendo al directorio de la demo y construyendolo con la siguiente instrucción:

```
$ ./build_all.sh >/dev/null
```

Tras esto se genera un archivo **.elf** que será la imagen del programa que se enviará al SOM y que a continuación se ejecuta en él. También se pueden encontrar distintas opciones para ejecutar un programa en el SOM, sin embargo, se decide desde Linux, parando el procesador inicialmente en el caso de que este encendido, cargando el *firmware* y finalmente activando el procesador.

```
# echo start >/sys/class/remoteproc/remoteproc0/state
```

Para probar que el comportamiento es adecuado se conecta un LED en el pin indicado en la demo y se observa un parpadeo que se enciende y apaga cada segundo.

Tras esto se realiza un proyecto en el que cuando se active una entrada se realice un cambio en una salida. Para ello a partir de la demo previamente mencionada, se realizará una versión con estas características.

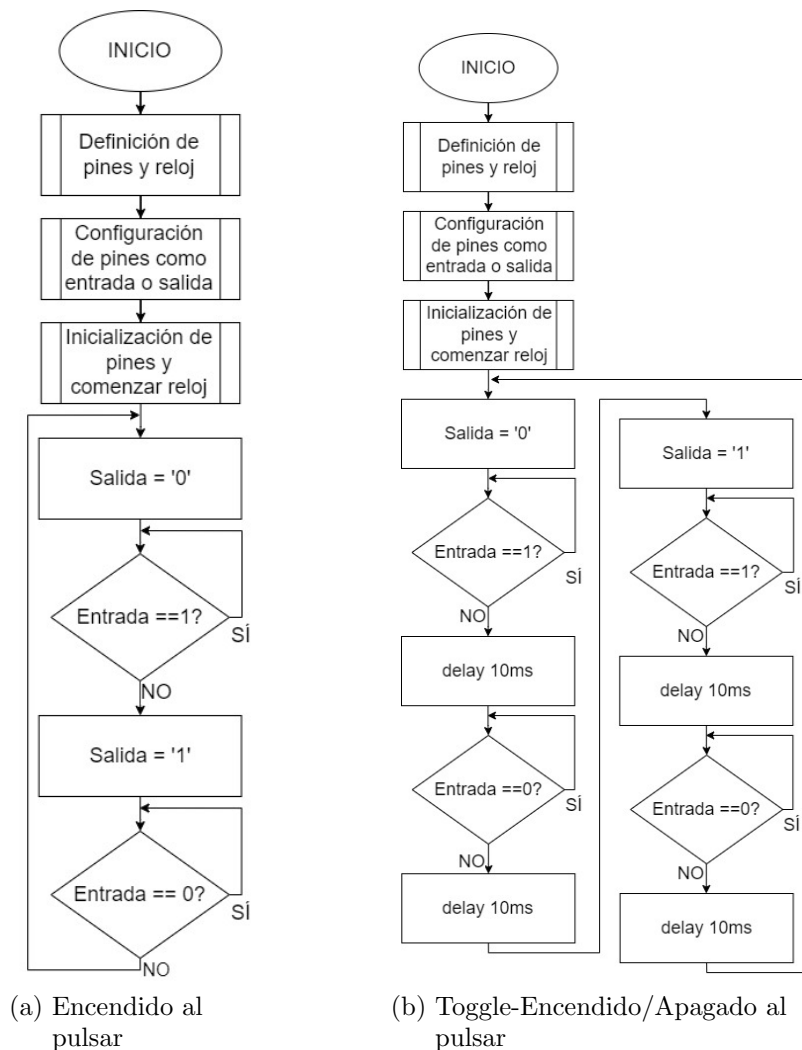


Figura 3.11: Activación de una salida a partir de una entrada con M-33

Para ello, con la estructura de la demo se modificarán determinados archivos .c y de configuración del proyecto que permitirán construirlo y posteriormente ejecutar la idea mencionada. Se ejecutan dos opciones que serán que la salida se active cuando se active la entrada y viceversa como en la Figura 3.11a, y por otro lado que cuando la entrada se active lo haga la salida pero si se vuelve a activar se apague como en la Figura 3.11b.

Primero, se configuran los pines deseados en el archivo de configuración, seleccionando si es necesario el uso de resistencias de pull-up o pull-down si es necesario y otras características. Posteriormente se escribirán los códigos principales siguiendo la estructura de los anteriores diagramas dependiendo de la opción escogida.

Finalmente, como se ha hecho en la demo, se construye el proyecto y se envía el firmware al SOM donde se cargará y ejecutará en el co-procesador.

Si se quisiera ver los cambios de estado tanto en esas entradas como salidas en el software de automatización utilizado se podría comunicar con el procesador principal con algún protocolo de comunicación como MQTT, por ejemplo, en el caso de que esto fuera necesario, sin embargo no se profundiza más en este sentido.

Capítulo 4

Software

4.1. Introducción

En este capítulo se va a realizar el análisis del software, comenzando por el estudio de su arquitectura, explicando como funciona, sus características y funcionamiento.

Posteriormente con el software detallado, se desarrollará y diseñará el software a aplicar en el entorno del proyecto.

Tras el análisis realizado en el capítulo 2, se llega a la conclusión que la mejor opción es **Home Assistant**, siendo el software sobre el que se va a desarrollar el trabajo. La información obtenida será principalmente de la web de Home Assistant[45]

4.2. Arquitectura del Software

4.2.1. Definición del Software

Home Assistant es un software de código abierto para automatización domótica con una alta valoración en el sector y con una comunidad mundial muy extensa, que permite la automatización de un entorno de forma sencilla e inteligente. Aprovechando la gran variedad de sensores e *integraciones* que incluye, aporta al usuario una elevada cantidad de posibilidades para el control del lugar en el que se encuentra.



Figura 4.1: Logotipo de Home Assistant

Se podría decir que Home Assistant es un *controlador*, es decir, un software que necesita de distinto hardware que se comunicará con este por medios tanto cableados como inalámbricos. Por ejemplo, un sensor toma una medida que será enviada al software, donde se podrá almacenar y tratar[46].

4.2.2. Características del Software

Como se puede apreciar en la Figura 4.2, Home Assistant es ampliamente elegido en el sector, teniendo una grata valoración y amplia comunidad que lo avala [47].

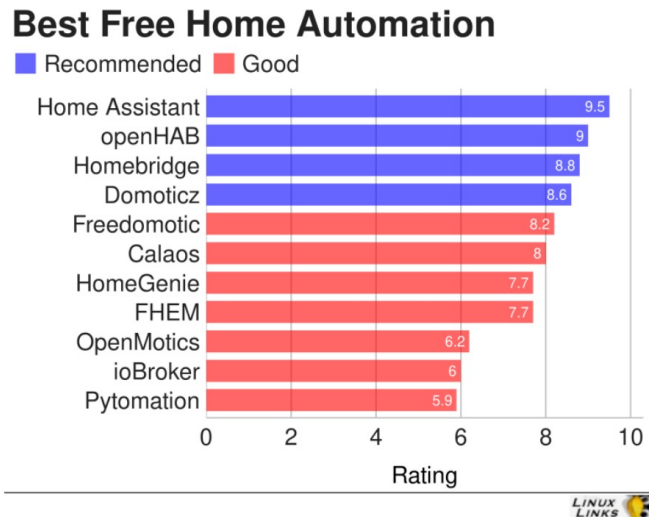


Figura 4.2: Mejores softwares gratuitos para domótica (Fuente: LinuxLinks)

Home Assistant incorpora más de 1900 *integraciones* que hacen referencia al amplio abanico de marcas que incluye, pudiendo acceder a gran cantidad de dispositivos y servicios diferentes.

Respecto a estas integraciones, permite el uso de distintos protocolos de comunicación como Modbus, WiFi, Bluetooth, Zigbee o MQTT entre otros. En referencia a este último, para configurarlo se elegirá un broker, que puede ser privado o público.

Se dispone adicionalmente de integración de diagnósticos, que permite al usuario acceso a reportes del comportamiento del software.

Home Assistant mantiene los datos localmente sin necesidad imperativa de una nube y pudiendo trabajar con este software de manera offline. El software se comunica de manera local con los dispositivos y extraerá datos de la nube si no hay otra opción, procesando todo localmente. Esto no impide el acceso a la nube, todo lo contrario, Home Assistant permite el uso y análisis de datos de la nube de forma práctica.

Al mismo tiempo, Home Assistant dispone de aplicación para dispositivos móviles que permite conocer ubicaciones de usuarios, notificar de acciones o acceder de manera remota al software entre otras características.

Home Assistant principalmente se utilizará de manera online para su acceso y control, sin embargo, en el caso de disponer de dispositivos que funcionen de manera offline o de encontrarse en una red local, se podrá hacer uso sin ningún inconveniente.

En la situación de necesitar acceso de manera remota, será viable mediante varios métodos, pudiendo usar Remote UI tomando acceso desde el servicio de *Home Assistant Cloud*. De la misma manera, se podrá utilizar una VPN.

4.2.3. Instalación del Software

Para la instalación del software se dispone de distintos métodos genéricos, siendo mejor una u otra opción posteriormente dependiendo del sistema operativo e interfaz sobre la que se va a instalar.

Home Assistant ofrece cuatro métodos, recomendando dos de ellos:

- ***Home Assistant Operating System***

Sistema Operativo mínimo optimizado para potenciar al software, incorporando un *Supervisor* que permite administrar *Home Assistant Core* y los *Add-ons*.

- ***Home Assistant Container***

Basada en contenedores de *Home Assistant Core* como por ejemplo *Docker*.

Por otro lado se encuentran Home Assistant Supervised y Home Assistant Core en los que la instalación es por línea de comandos, tanto del Supervisor en la primera como del núcleo en la segunda.

Home Assistant recomienda Home Assistant Operating System debido a las ventajas que se pueden observar en la siguiente Figura 4.3.

	OS	Container	Core	Supervised
Automations	✓	✓	✓	✓
Dashboards	✓	✓	✓	✓
Integrations	✓	✓	✓	✓
Blueprints	✓	✓	✓	✓
Uses container	✓	✓	✗	✓
Supervisor	✓	✗	✗	✓
Add-ons	✓	✗	✗	✓
Backups	✓	✓ ¹	✓ ¹	✓
Managed Restore	✓	✗ ²	✗ ²	✓
Managed OS	✓	✗	✗	✗

Figura 4.3: Comparativa de métodos de instalación (Fuente: Home Assistant)

Según el dispositivo y plataforma en el que se utilice este software, se permite la instalación en distintos Sistemas Operativos como Windows, Linux, macOS, etc. Para instalar *Home Assistant Operating System* basándose en un entorno Linux, se creará una máquina virtual y se configurará con una imagen de Home Assistant. Finalmente, se ejecuta la máquina y cuando el arranque finalice se podrá acceder a la interfaz web del software.

4.2.4. Funcionamiento y uso del Software

La interfaz web de Home Assistant es muy completa y de uso intuitivo basada en *Dashboards*. Estas, son páginas personalizables que muestran la información, son rápidas y son una forma potente de manejo del entorno usando móviles u ordenadores.

Por otro lado, se dispone de *Dispositivos* y *Entidades*. Respecto a los primeros, son una agrupación de una o más entidades, pueden representar un dispositivo físico con

uno o más sensores que aparecerán como entidades asociadas. En los Dashboards se podrá ver su estado, y por otra parte, se pueden utilizar en diferentes *Automatizaciones*.

Como se acaba de mencionar Home Assistant permite programar mediante *Automatizaciones* o *Scripts* y configurando dispositivos mediante *YAML* que es el lenguaje principal para configuraciones. Sin embargo, además de este lenguaje, se podrá utilizar *Python*, lenguaje en el que está escrito en sí mismo Home Assistant.

Las *Automatizaciones* son un conjunto de acciones repetibles, configurables para que se ejecuten automáticamente. Compuestas de: *disparadores/triggers* que inician la automatización, *condiciones* para continuar, y *acciones*.

Del mismo modo, los *Scripts* también son acciones repetibles. La diferencia respecto a las Automatizaciones es que estos no tienen disparadores, no pudiéndose ejecutar automáticamente de no ser que estén dentro de una automatización. Serán útiles para simplificar automatizaciones en las que se repiten procesos.

También se podrán crear configuraciones predefinidas con diferentes dispositivos configurados a la vez en una *Escena*, sin tener que configurarlos individualmente.

4.3. Diseño del Software

En esta sección se va a realizar el diseño y del software a desarrollar, que será Home Assistant, argumentando las tomas de decisiones y su procedimiento.

4.3.1. Instalación de Home Assistant

En el apartado previo se han señalado y explicado distintos métodos de instalación de Home Assistant. En este caso, el hardware utilizado se basa en un Sistema Operativo Linux. Por otra parte, se mostraban cuatro alternativas para la instalación, y basándose de igual forma en el hardware utilizado, la interfaz que ofrece es básicamente textual, lo que complica en determinada manera la instalación rápida de métodos como *Home Assistant Operating System* o *Home Assistant Container*, teniendo que acceder a comandos distintos para llevarlos a cabo.

Es por ello que en una primera instancia, se procede a instalar ***Home Assistant Core*** siguiendo los pasos indicados en la web de Home Assistant [45].

Para la instalación de Home Assistant Core en primer lugar el sistema tiene que estar actualizado para poder instalar los paquetes necesarios respecto a la instalación. Con el sistema actualizado a la hora de instalar las dependencias se encuentra un problema con una de ellas, esto se puede solucionar no instalando ese paquete que posteriormente no interferirá en la instalación, no siendo por tanto importante.

Home Assistant sugiere crear un usuario de homeassistant. Sin embargo, si se quiere acceder a distintos periféricos de la SBC o características necesarias, es importante que el usuario activo sea el administrador del sistema ***root***.

Por tanto, no se deberá crear y activar el usuario *homeassistant*, sino que se crea el directorio de instalación de homeassistant y se pasa a este mediante el usuario *root*. Tras ello se crea y activa el entorno virtual.

```
# python3 -m venv .
```

```
# source bin/activate
```

Una vez el entorno virtual ha sido activado la línea de comandos se verá como: (homeassistant) root@imx93-var-som:/srv/homeassistant # tan solo quedando instalar el paquete Python requerido para la instalación de Home Assistant Core. Una vez este se haya completado ya sí que se podrá instalar la versión de Home Assistant Core.

```
# pip3 install homeassistant==2023.8.3
```

Finalmente se da comienzo a Home Assistant Core, completando de esta forma la instalación y creando un directorio de configuración y permitiendo poder acceder a la interfaz web de Home Assistant:

```
# hass
```

Home Assistant Core es una buena versión, sin embargo tiene una serie de defectos. En primer lugar, no permite el uso de complementos o Add-ons, que son muy útiles y hacen que Home Assistant sea más completo y sencillo de utilizar. No obstante, como mayor defecto, se encuentra que para realizar configuraciones o crear automatizaciones será necesario modificar los archivos configuration.yaml y otros que se encuentren en el directorio de configuración, siendo fundamental acceder mediante el host Putty al contenido de estos, y haciendo que el procedimiento sea algo tedioso.

Por ello, se decide instalar la versión similar pero más completa **Home Assistant Supervised**. Aunque su proceso de instalación sea algo más complejo, su resultado es más completo y agradable para un uso posterior.

En primer lugar, este método tiene requerimientos muy estrictos, tan solo soportando Debian en el caso de Linux. En este caso, al estar usando un S.O. Debian esta instalación es posible. Con esto claro, se accede al proceso de instalación [48]. El supervisor es un dispositivo completo que administrará todo el sistema. El usuario será responsable de que los componentes necesarios estén instalados y mantenidos.

Se comienza instalando las distintas dependencias necesarias y una vez estas se han completado será necesario instalar un entorno de desarrollo de contenedores Docker-CE. Tras ello, el siguiente paso será instalar el *OS-Agent*, que es la forma en la que el supervisor maneja tareas adicionales en el host [45]. Para ello, se pueden encontrar distintos agentes dependiendo de arquitecturas de procesadores descargando el adecuado a la arquitectura de este, que como se comprueba es de tipo **aarch64** . [49]

Tras la descarga, el archivo se copiará desde el PC en el que se ha descargado hasta el SOM y se instala ahí. Finalmente será necesario instalar el paquete Debian de Home Assistant Supervised.

```
# sudo dpkg -i os-agent_1.0.0_linux_aarch64.deb
```

```
# wget https://github.com/home-assistant/supervised-installer/releases/latest/download/homeassistant-supervised.deb apt install ./homeassistant-supervised.deb
```

4.3.2. Comenzando con Home Assistant

Con Home Assistant Supervised instalado se reinicia la SBC y durante el arranque se iniciará el *Supervisor*.

Con el arranque completado se accede a la interfaz web de Home Assistant introduciendo la dirección ***http://<IP_ADDRESS>:8123***, siendo en este caso la IP asignada al módulo. En el primer acceso se configurará la cuenta a utilizar, pudiendo acceder posteriormente introduciendo el nombre de usuario y la contraseña.

Dentro de la interfaz se podrán visualizar las distintas tarjetas personalizables, acceso a los ajustes necesarios, diseño de las automatizaciones y configuración de las integraciones.

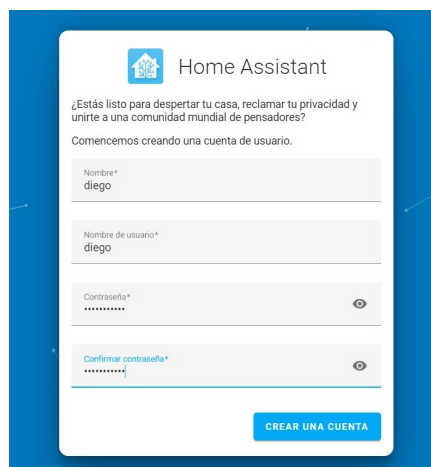


Figura 4.4: Configuración inicial de Home Assistant

4.3.3. Interfaz y ajustes de Home Assistant

Home Assistant ofrece una tarjeta de inicio personalizable en la que se puede indicar el estado de distintos sensores, activar o no algún interruptor o ver el estado del entorno mediante alguna gráfica o esquema.

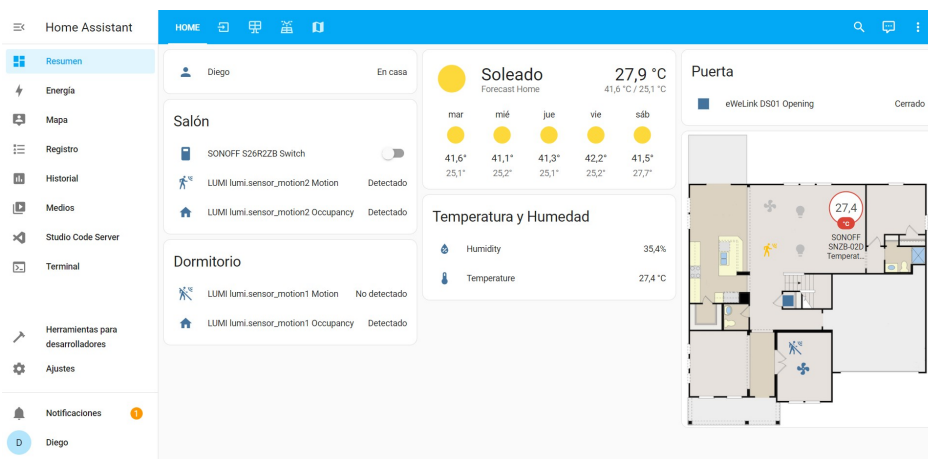


Figura 4.5: Tarjeta de control y visualización del entorno domótico

Como se puede apreciar en la Figura 4.5, esta tarjeta puede ser editada al gusto del usuario, en la parte superior se pueden observar distintos iconos que hacen referencia a varias páginas creadas y que igual que la actual pueden personalizarse.

Por otro lado, en la parte izquierda se encuentran las distintas opciones de usuario, dando acceso a la tarjeta de Resumen que es la inicial o al Registro que muestra los últimos procesos realizados y posibles fallos entre otras. Home Assistant Supervised permite la instalación de complementos que también pueden visualizarse en esta barra

de herramientas, como pueden ser *Studio Code Server* que permitirá configurar dispositivos y programar automatizaciones de manera rápida y cómoda, o la terminal SSH que facilitará el acceso a esta desde la propia interfaz web.

4.3.4. Configuración de sensores, integraciones y comunicaciones

Antes de programar automatizaciones, será necesario configurar los distintos dispositivos o en su defecto comunicaciones que permitan manejar de manera adecuada el entorno. Home Assistant ofrece distintas opciones, desde la búsqueda de integraciones ya generadas en el propio software, o por otro lado, en el caso de comunicaciones o dispositivos sin una configuración definida por Home Assistant, es posible la creación de estos sensores o dispositivos mediante los archivos de configuración de Home Assistant.

Se comenzará por las integraciones sencillas como las de sensores Zigbee y posteriormente se configurarán el módulo de relés como otras distintas comunicaciones mediante Studio Code Server.

4.3.4.1. Sensores Zigbee

En primer lugar, para realizar la comunicación Zigbee es necesario un dispositivo que funcione como router y que permita enlazar los distintos sensores, interruptores o pulsadores Zigbee. Para ello, se dispone del *Zigbee 3.0 USB Dongle Plus* que servirá como punto de unión Zigbee. Se comienza conectándolo a la Symphony Board, posteriormente en Home Assistant se accede a la página de integraciones y entidades. Una vez ahí, se abrirá la opción **Añadir Integración**, y en ella se buscará la integración **Zigbee Home Automation (ZHA)**.



Figura 4.6: Integración de Zigbee Home Automation y ZBDongle-E

Tras elegirla, será necesario indicar el puerto serie adecuado para su configuración. Si el dispositivo es encontrado automáticamente a través del hardware, se continúa eligiendo ese puerto y se indica el tipo de radio[50]. Con la puerta de enlace integrada, se accede a ella apreciando la interfaz de la Figura 4.6. En ella se pueden buscar e integrar los distintos dispositivos y sensores Zigbee. Si estos se encuentran cerca y en *modo emparejamiento*, entonces aparecerán en pantalla permitiendo integrarlos.

Con los dispositivos configurados se tendrá posibilidad de acceder a ellos de forma general como a sus entidades (características de cada sensor) pudiendolos añadir a la tarjeta de Resumen.

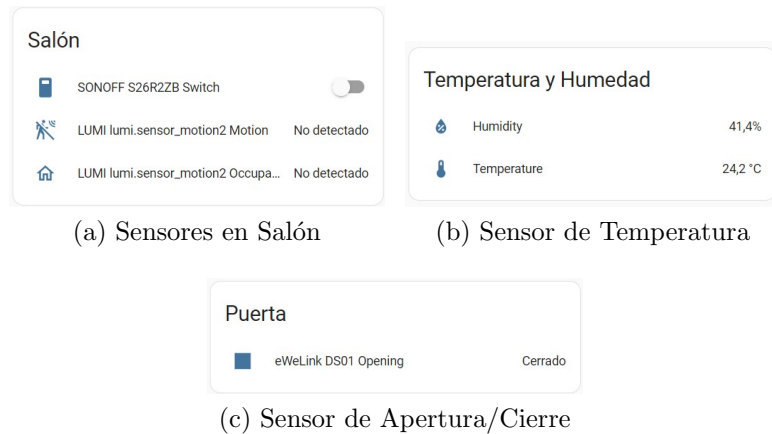


Figura 4.7: Uso de sensores Zigbee tras su configuración

Este procedimiento se puede extrapolar a cualquier otra integración disponible en Home Assistant. Este proceso se podría usar también para configurar **Bluetooth**, y como el SOM ya incluye este tipo de comunicación, será detectada automáticamente.

4.3.4.2. Módulo de Relés Dingtian

El módulo de relés permite una amplia variedad de opciones para poder controlar mediante su ESP32 las distintas entradas y relés de los que dispone. A pesar de que no se encuentra entre las integraciones habilitadas por Home Assistant, permite comunicarse con este software mediante **MQTT**. [43]

Por tanto, hará falta comunicar vía MQTT el SOM con el módulo de relés, siendo necesario un broker que facilite esta conexión. Se instala de este modo un broker **mosquitto** en el SOM. Existen dos opciones, a partir de un complemento en Home Assistant o mediante la terminal Putty. Por seguridad de comunicación entre el SOM y el módulo se decide la segunda opción. En primer lugar, se comprueba que el sistema esté actualizado completamente y a continuación se instala y habilita mosquitto: [51]

```
# sudo apt install mosquitto
# sudo systemctl enable mosquitto
```

Con el broker instalado, se configuran en la interfaz del módulo de relés los parámetros para la comunicación MQTT, siendo la dirección del broker la del SOM, como el número de puerto entre otros.

El siguiente paso será escribir el código que consiga configurar en Home Assistant las distintas entradas y relés del módulo. Para ello, a través del complemento Studio Code Server, se crea un archivo **dingtian.yaml** dentro del directorio de configuración. Se decide que sea de tipo *yaml* debido a que es un lenguaje de programación derivado de Python que ofrece un sistema de configuración de dispositivos muy intuitivo, sencillo y rápido.

Basandose en código MQTT que el propio Dingtian provee en su manual de usuario y adaptaciones realizadas para su lectura en Home Assistant, se programa la comunicación de la siguiente manera:

```

mqtt:
  switch:
    - unique_id: Dingtian-relay20379-r1
      name: "Dingtian-relay20379 switch_1"
      state_topic: "/dingtian/relay20379/out/r1"
      command_topic: "/dingtian/relay20379/in/r1"
      availability:
        - topic: "/dingtian/relay20379/out/lwt_availability"
          payload_available: "online"
          payload_not_available: "offline"
      payload_on: "ON"
      payload_off: "OFF"
      state_on: "ON"
      state_off: "OFF"
      optimistic: false
      qos: 0
      retain: false
    - unique_id: Dingtian-relay20379-r2
      name: "Dingtian-relay20379 Switch2"
      state_topic: "/dingtian/relay20379/out/r2"
      command_topic: "/dingtian/relay20379/in/r2"
      availability:
        - topic: "/dingtian/relay20379/out/lwt_availability"
          payload_available: "online"
          payload_not_available: "offline"
      payload_on: "ON"
      payload_off: "OFF"
      state_on: "ON"
      state_off: "OFF"
      optimistic: false
      qos: 0
      retain: false

```

(a) Activación de relés en HA

```

binary_sensor:
  - unique_id: Dingtian-relay20379-i1
    name: "Dingtian-relay20379 Input1"
    state_topic: "/dingtian/relay20379/out/r1"
    availability:
      - topic: "/dingtian/relay20379/out/lwt_availability"
        payload_available: "online"
        payload_not_available: "offline"
    payload_on: "ON"
    payload_off: "OFF"
    qos: 0
  - unique_id: Dingtian-relay20379-i2
    name: "Dingtian-relay20379 Input2"
    state_topic: "/dingtian/relay20379/out/r2"
    availability:
      - topic: "/dingtian/relay20379/out/lwt_availability"
        payload_available: "online"
        payload_not_available: "offline"
    payload_on: "ON"
    payload_off: "OFF"
    qos: 0

```

(b) Lectura estado de relés

```

  - unique_id: Dingtian-entrada1
    name: "Dingtian-entrada1"
    state_topic: "/dingtian/relay20379/out/i1"
    availability:
      - topic: "/dingtian/relay20379/out/lwt_availability"
        payload_available: "online"
        payload_not_available: "offline"
    payload_on: "ON"
    payload_off: "OFF"
    qos: 0
  - unique_id: Dingtian-entrada2
    name: "Dingtian-entrada2"
    state_topic: "/dingtian/relay20379/out/i2"
    availability:
      - topic: "/dingtian/relay20379/out/lwt_availability"
        payload_available: "online"
        payload_not_available: "offline"
    payload_on: "ON"
    payload_off: "OFF"
    qos: 0

```

(c) Lectura estado de las entradas

Figura 4.8: Comunicación MQTT entre el módulo de relés y Home Assistant

En la Figura 4.8a se puede observar la configuración mqtt de interruptores MQTT. A partir de esta configuración se generarán distintos switch en la interfaz de HA con los que se podrá activar los distintos relés. Para esta configuración MQTT se añaden una id única con la que identificar el dispositivo; **state_topic** y **command_topic**, con los que se indicará el tema MQTT en el que se publicará el estado actual del switch y el tema en el que se publicarán los comandos para controlarlo respectivamente. **Availability** proporcionará la información sobre la disponibilidad del dispositivo. **Payload_on** y **payload_off** definirán los mensajes a enviar en el tema de comando, **state_on** y **state_off** darán los valores esperados del estado del relé. **Optimistic: false** hará que el interruptor espere a la confirmación del dispositivo antes de cambiar de estado, **qos** indica el nivel de calidad de servicio de los mensajes enviados y con **retain: false** no se guardarán los mensajes de estado en el broker. Esta configuración se realizará de forma repetida y adaptada a cada uno de los relés.

Respecto a la Figura 4.8b y la Figura 4.8c, se configura, en el caso de la primera, el estado de los relés, y para la segunda, el estado de las entradas digitales. Tendrán la misma configuración que los interruptores creados anteriormente, a diferencia de que en este caso no hará falta especificar el **command_topic** ni por tanto **state_on** y **state_off**. Como en el caso de los interruptores, se repetirá la configuración de cada sensor binario para cada uno de los restantes relés y entradas.

Cabe mencionar que tras escribir el código de configuración de los relés, estos no se encontraban disponibles, no encontrando ningún fallo a simple vista, con el broker y conexión con el módulo correctas. Por lo tanto, se prueba a utilizar *MQTT Explorer* que es un cliente genérico MQTT y tras introducir los datos del broker en este, se muestra correcta la comunicación de Home Assistant con el módulo de relés.

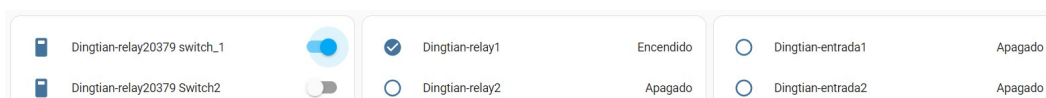


Figura 4.9: Activación y lectura de relés en Home Assistant

4.3.4.3. Modbus

El entorno dispone de distintos equipos que para su análisis es necesario realizar comunicación Modbus. Home Assistant sí que dispone de una integración Modbus incluida, sin embargo, para ser usada tiene que ser configurada mediante un lenguaje de programación.

De la misma forma que para MQTT, se programa la integración Modbus en *Studio Code Server* con YAML. Se configurará directamente esta integración en el archivo ***configuration.yaml***. En primer lugar, será necesario conocer el tipo de comunicación Modbus a utilizar, el host del dispositivo y el número del puerto. Con ello, se podrán configurar los distintos sensores (registros del dispositivo). La configuración Modbus se realizará de la siguiente manera:

```
modbus:
  - name: Ingeteam
    type: tcp
    host: howlab.ddns.net
    port: 502
    sensors:
      - name: PV1_Voltage
        scan_interval: 60
        unit_of_measurement: V
        slave: 1
        address: 31
        count: 1
        input_type: input
        data_type: uint16
        scale: 1
        precision: 2
```

Figura 4.10: Configuración Modbus de un inversor fotovoltaico

En la Figura 4.10 se encuentra la configuración de la tensión en en el *PV1* o *campo 1* de un ***inversor fotovoltaico***. Se comienza aportando los datos de conexión con el dispositivo Modbus que son los mencionados previamente.

Con el dispositivo ya configurado puede a determinar sus sensores / registros más interesantes para el análisis. Para integrar un registro del dispositivo como el de la anterior figura, es necesario determinar un nombre para identificarlo, un intervalo de escaneo en el que tras un determinado tiempo se tome una nueva muestra (en segundos). Se indicará la unidad de medida, el número de dispositivo esclavo con el que se desea comunicar, la dirección del registro que se desea leer (en la hoja de datos del inversor se indica que los registros van del 30001 al 30073, sin embargo para su configuración en Home Assistant realmente van del 0 al 72, por lo que el registro 30032 será el 31)[52]. Posteriormente, se indica la cantidad de registros a leer, siendo 1 si el tipo de datos es uint16, 2 para uint32 o 4 para uint64. También se debe indicar el tipo de entrada para poder leer correctamente la información. Finalmente, se especifican la escala del valor a mostrar y la precisión de lectura (número de decimales a leer).

Tras incluir esta configuración, será posible leer el valor del registro en Home Assistant desde su respectiva entidad. Este procedimiento de lectura del sensor se repetirá para los registros más importantes, con el fin de analizar el dispositivo (en el caso del inversor tensiones, corrientes y potencias de distintos campos, batería y entregadas a la red o a equipos del entorno).[52]

Este proceso de configuración de Modbus, será el que se realice con cualquier otro equipo que necesite de protocolo Modbus como pueden ser un contador de la luz, un

aire acondicionado A/C o un calentador de agua. Esto se puede ver a continuación en la Figura 4.11a, en la Figura 4.11b o en la Figura 4.11c.

En el contador se pueden obtener parámetros relevantes como pueden ser la tensión, corriente, potencia, energía de cada fase y la frecuencia de la red. [53] Respecto al aire acondicionado, se podrán escribir registros como iniciar los grupos, activar distintos modos o elegir la posición de las paletas. [54] Finalmente, en cuanto al calentador de agua se pueden leer registros como el flujo de temperatura, temperatura de la sala, humedad o estado del circuito calentador. [55]

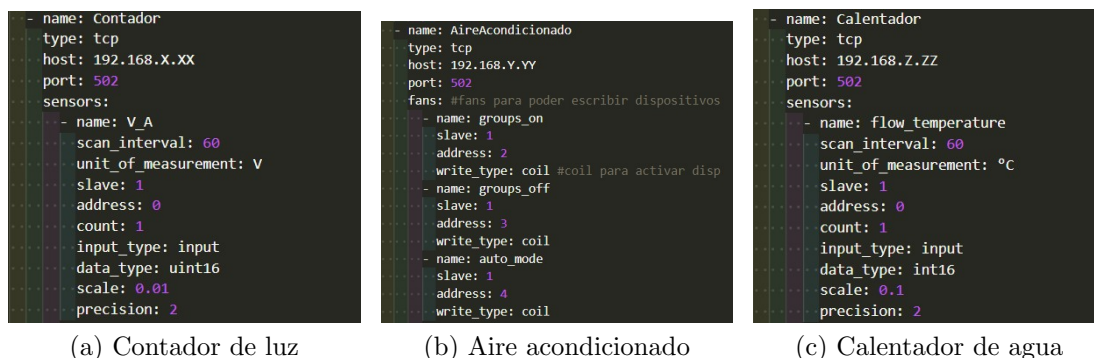


Figura 4.11: Configuración de otros dispositivos modbus

4.3.4.4. Acceso remoto y aplicación móvil

Para finalizar este capítulo cabe mencionar el posible acceso remoto al software, acceso a la nube y el posible uso de Home Assistant.

Respecto a la comunicación con la nube, Home Assistant permite un servicio llamado **Home Assistant Cloud** en el que creando una cuenta en *Nabu Casa* permite acceder de manera remota a Home Assistant mediante una URL, a su vez ofrece el uso de asistentes de voz como pueden ser Google Assistant o la asistente de Amazon *Alexa*. Al mismo tiempo permite el acceso a otros servicios que requieren de comunicación con la nube como pueden ser, por ejemplo, los pronósticos del tiempo.

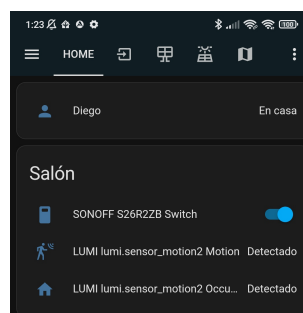


Figura 4.12: Acceso a Home Assistant desde un dispositivo móvil

Home Assistant permite acceso al software a través de un dispositivo móvil, aportando flexibilidad y permitiendo usar los sensores de este dispositivo, conocer su ubicación o poder acceder remotamente al software.

4.3.5. Diseño de automatizaciones

Teniendo ya todo a punto para poder ser utilizado, se puede comenzar a realizar las distintas automatizaciones oportunas.

Para ello, se va a realizar una serie de automatizaciones en las que se implicará tanto a los sensores Zigbee como a la placa de relés, teniendo siempre en cuenta la posible actuación de las personas de manera voluntaria en el entorno, por ejemplo, activando un pulsador/interruptor de manera manual sin tener en cuenta estas automatizaciones.

El proceso de diseño de la automatización es el siguiente: En primer lugar se realizarán distintos diagramas de flujo, posteriormente se programará en Python la mayor parte de la automatización, sin embargo, el motivo de arranque o también llamado **trigger** se programará en el propio sistema de configuración de automatizaciones de Home Assistant.

4.3.5.1. Diagramas de flujo

Por lo tanto se va a comenzar esquematizando las automatizaciones que se quieren realizar. En primer lugar, y como prueba del funcionamiento de este tipo de automatizaciones, se deseará que si hay movimiento detectado o no se haya dejado de detectar por 2 minutos, se encienda un interruptor, teniendo en cuenta que no lo apague una persona manualmente.

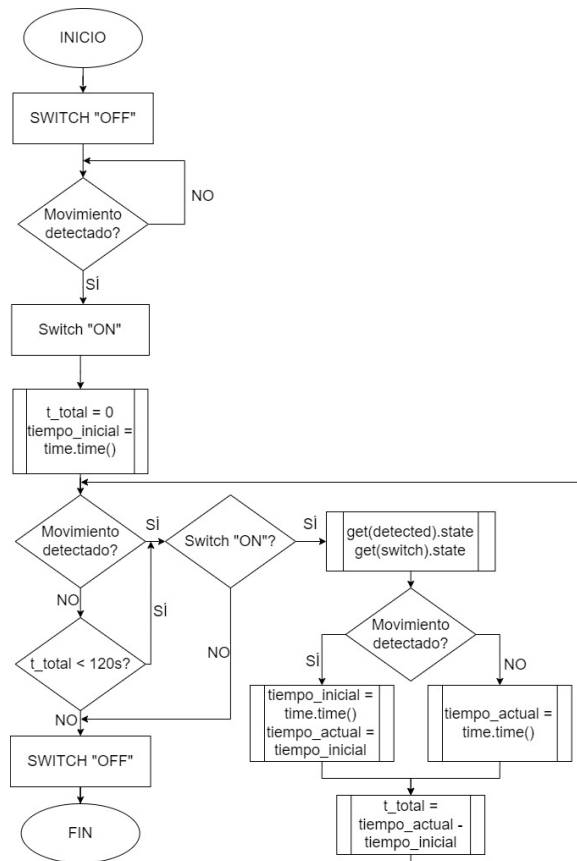


Figura 4.13: Automatización de ejemplo del switch

A partir de la anterior automatización, se realiza una adaptación que sirve para encender o apagar un ventilador o A/C en verano o en el caso de invierno se trataría de una calefacción.

Si se detecta movimiento comienza la automatización. Para el caso de un ventilador, Figura 4.14a, tomando como referencias de temperatura 27°C y 28°C, si la temperatura

supera los 28°C (en verano), se encenderá el ventilador o A/C. Mientras siga por encima de 27°C, y se detecte movimiento (o no pero no hayan pasado 120 segundos), continuará encendido, teniendo en cuenta que no lo haya apagado alguien manualmente.

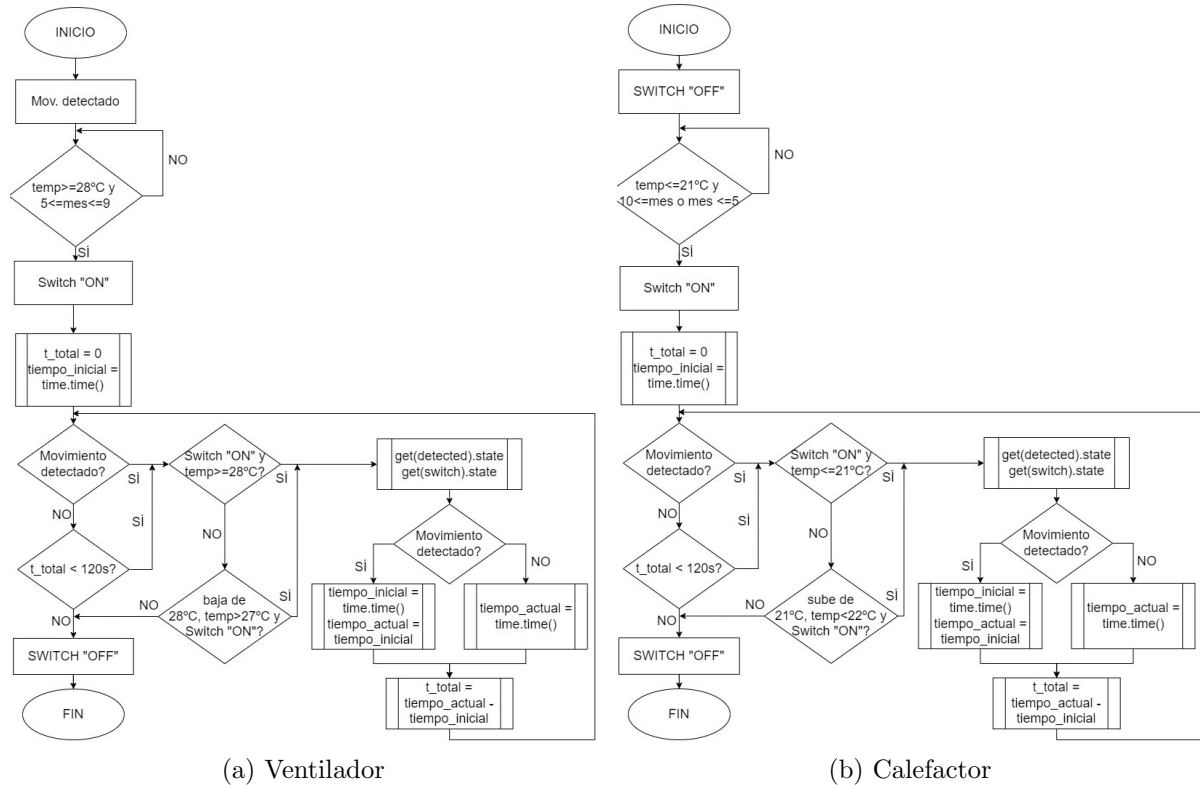


Figura 4.14: Automatizaciones de distintos dispositivos climatizadores

El caso de la calefacción Figura 4.14b, es similar, tratándose ahora de invierno y en el que mientras la temperatura sea inferior a 21°C se encenderá la calefacción, manteniéndose hasta alcanzar un máximo de 22°C, cuando se apagará hasta volver a bajar de 21°C. Todo ello siguiendo el mismo procedimiento de detección de movimiento.

A continuación, se va a realizar una nueva automatización en la que se desea activar distintos dispositivos dependiendo de la acción realizada con el pulsador Zigbee.

El pulsador dispone de tres acciones, pulso corto, doble pulso o pulso sostenido. Para simular los dispositivos se utilizará como salidas los relés del módulo Dingtian, sin embargo estas salidas pueden ser cualquier luz del entorno, ventilador u otro dispositivo.

Si una persona activa el pulsador con un solo pulso, Figura 4.15a, se encenderán las luces 1 y 2. Se supone que para poder usar este pulsador en cualquier caso un sensor de movimiento ha debido detectar movimiento en esa misma sala previamente. Las luces se mantendrán encendidas mientras se detecte ocupación y no se hayan apagado manualmente. Este mismo proceso ocurrirá con las luces 3 y 4 en el caso de que se realice un doble pulso, Figura 4.15b, o con las luces 5 y 6 si se mantiene pulsado, Figura 4.15c.

Finalmente se realiza una automatización que si detecta que se ha abierto la puerta en distintas circunstancias enciende o apaga las luces.

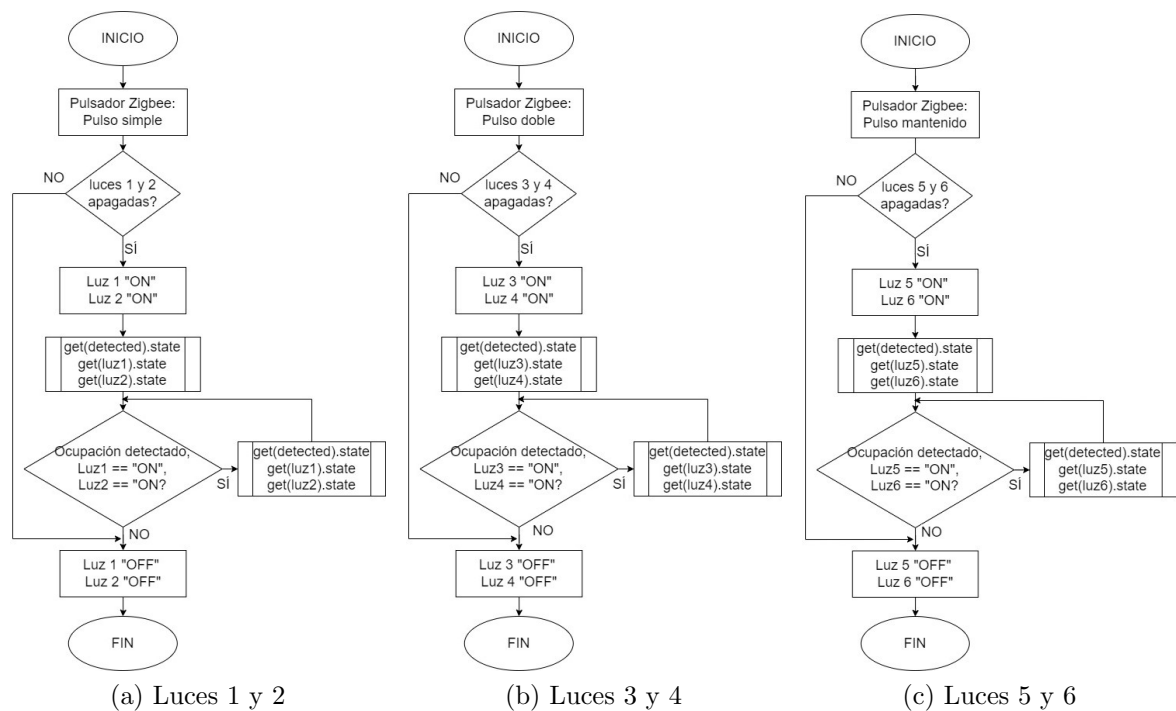


Figura 4.15: Automatización de luces dependiendo del pulso del pulsador Zigbee

Como se puede ver en el siguiente diagrama, si la persona desea salir las luces estarán encendidas hasta que se cierre la puerta. Por otro lado, si esa persona va a entrar, al abrir se detecta movimiento y se encienden las luces. Estas se apagaran cuando se deje de detectar movimiento o se apaguen manualmente.

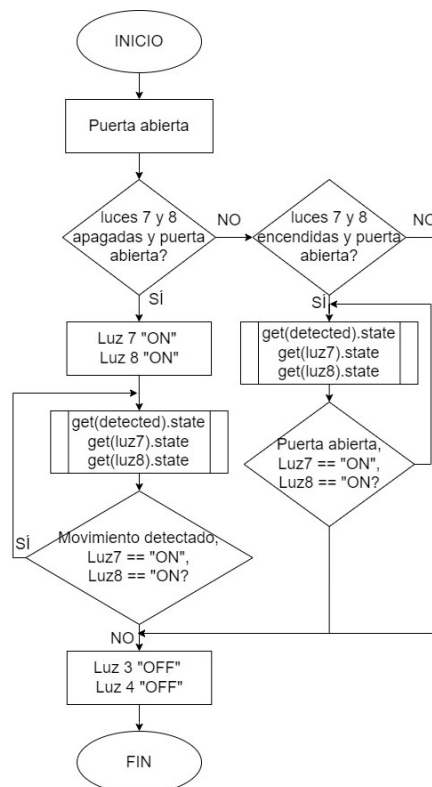


Figura 4.16: Automatización de apertura/cierre de una puerta

El sensor de apertura podría utilizarse como condición de apertura o no de ventanas en los programas del ventilador y el calefactor, en el caso de que la ventana este abierta se apagarían tales dispositivos. Por otro lado, en esta última automatización se podría tener en cuenta el momento horario para que no se encendan las luces en el caso de horas con bastante luz, o usar algún sensor que detecte el nivel de luz, no encendiendo la luz si hay más que un cierto umbral. No obstante, no se han realizado estas aplicaciones.

4.3.5.2. Programación de las automatizaciones

Para la programación de las anteriores automatizaciones se escribirá el código en Python activando los scripts de Python en el archivo de configuración.

Una vez creado el script correspondiente a una automatización, se accederá a través de la interfaz de Home Assistant a su apartado de Scripts y Automatizaciones. Se elige **Crear Automatización** y en ella se asignará un *trigger* que la active, cuando este suceso se cumpla, se llamará a el Python.Script adecuado dependiendo de la automatización a realizar.

La programación de estas automatizaciones puede realizarse mediante yaml, sin embargo, es más intuitivo y rápido el *editor visual* en el que tan solo hará falta indicar la entidad que lance la automatización, y como acción realizar la llamada al Script de python adecuado.



Figura 4.17: Automatización de la puerta mediante editor visual

Capítulo 5

Análisis de la información obtenida a partir de los resultados

5.1. Introducción

En este capítulo se van a comentar los resultados de la ejecución de las distintas automatizaciones y se van a analizar los datos obtenidos a través del inversor fotovoltaico por la comunicación Modbus y como influiría el uso de paneles en una instalación.

5.2. Resultados y características de las automatizaciones

Tras realizar las distintas automatizaciones, en un principio se comenzó programando puramente con el editor visual de Home Assistant sin utilizar scripts escritos en Python. Aunque el funcionamiento pudiera ser el mismo realizando las mismas acciones, ocurre un problema en el primer caso si se encuentra en funcionamiento una automatización con un bucle en su interior. Este problema es que acapara el uso completo de uno de los cores del i.MX93 gastando muchos recursos, esto no ocurre mediante el código en Python, por lo que se benefician de recurrir a estos scripts. Mediante el comando **top** en el host Putty se puede apreciar esta diferencia.

VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
582524	342892	40800	S	50.3	16.9	2:47.25	python3
101144	23408	4656	S	5.0	1.2	44:57.34	pulseau+
111408	72724	71432	S	3.3	3.6	0:42.27	systemd+

VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
101144	23408	4656	S	2.3	1.2	45:16.94	pulseau+
11712	4132	2216	R	0.7	0.2	0:00.14	top
1344596	32552	7852	S	0.3	1.6	2:16.72	contain+

(a) Automatización sin Python

(b) Automatización con Python

Figura 5.1: Comparación de recursos usados en la automatización del ventilador

Es por ello que se toma la opción con Python a utilizar. Por otra parte, al ejecutar las automatizaciones con los distintos dispositivos Zigbee se comprueba que correctamente funcionan y que estos se enlazan adecuadamente. En la Figura 5.2a se muestra un esquema en el que se simulan los sensores y se aprecia como cuando la puerta se abre las luces se encienden al detectar movimiento. Por otro lado en la Figura 5.2b se muestra que con detección en la sala y habiendo pulsado tanto una vez como manteniendo pulsado el pulsador Zigbee se encienden las luces relativas a dichas acciones.

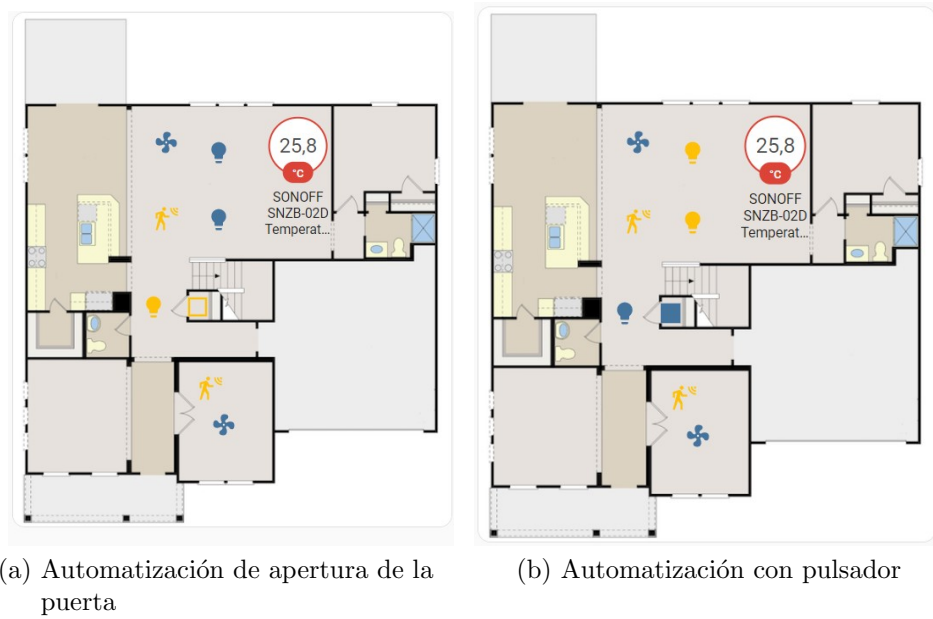


Figura 5.2: Resultados de la automatizaciones en un mapa de Home Assistant

5.3. Resultados del análisis del inversor Modbus

Tras la configuración del inversor mediante Modbus en Home Assistant se pueden tomar algunos datos bastante relevantes. En primer lugar, tanto la potencia como la corriente siguen la misma forma durante el tiempo, debido a que la tensión generada por el inversor es prácticamente constante durante el tiempo en el que el sol incide de manera efectiva sobre los paneles. En el momento que cae el sol la tensión se reduce considerablemente y la corriente y la potencia van disminuyendo hasta alcanzar los 0A y 0W respectivamente.

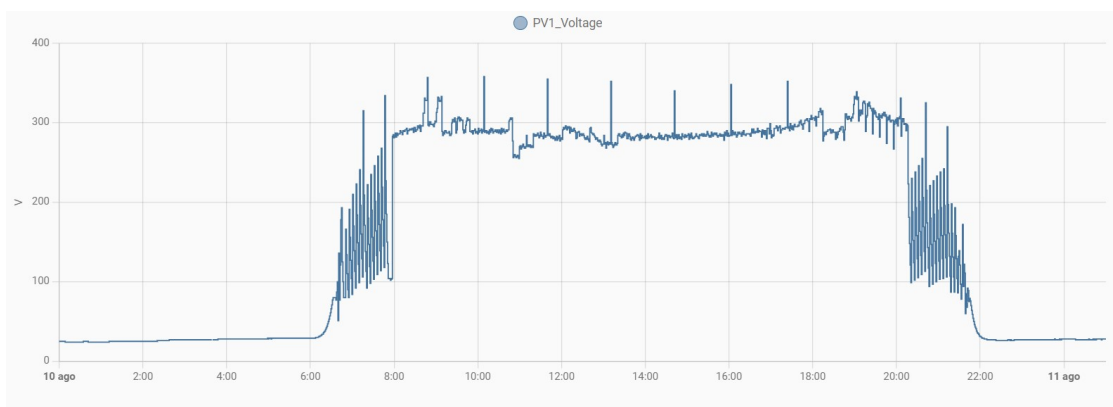


Figura 5.3: Tensión del campo A (V)

Es notable en esta gráfica de tensión, que el sol durante el momento en el que se ha tomado la gráfica (verano), empieza a salir de las 6:00 hasta las 8:00, momento en el que la tensión alcanza un máximo que mantiene más o menos constante (alrededor de los 300V). Por contrario, la luz dejará de incidir de manera progresiva desde las 20:00 hasta las 22:00 horas cuando la tensión será muy reducida (alrededor de 25V).

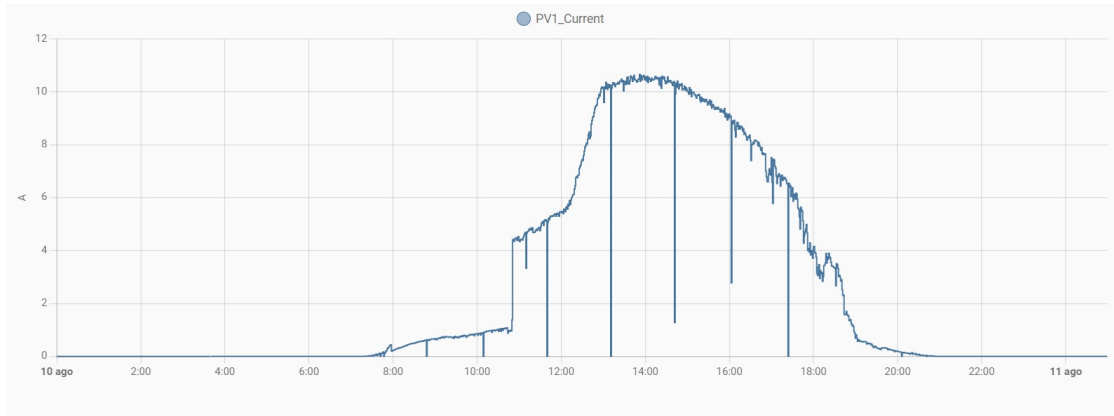


Figura 5.4: Corriente del campo A (A)

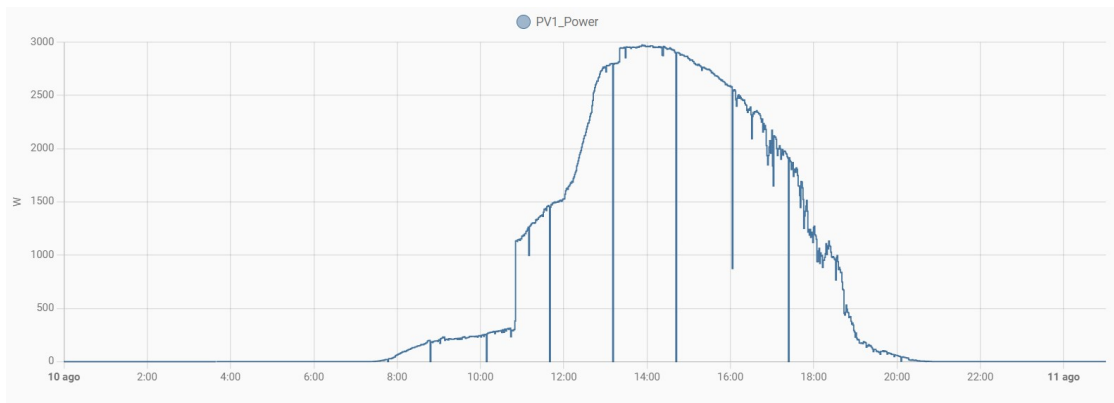


Figura 5.5: Potencia del campo A (W)

Las gráficas de corriente y potencia como se ha mencionado siguen una forma a lo largo del tiempo similar, debido a la continuidad de la tensión en prácticamente el mismo valor. Hasta alcanzar el medio día, tanto la corriente como la potencia irán aumentando su valor hasta un máximo, momento del día en el que habrá mayor incisión de los rayos del sol en el panel. A partir de ese momento irán reduciéndose sus valores hasta llegar a 0A y 0W.

Respecto al campo B la forma de las señales es similar a las del campo A, aunque con valores más reducidos ya que se utilizan de manera secundaria. Respecto a otros valores interesantes, se puede apreciar como efectivamente la frecuencia tanto en la corriente entregada a la red como a los dispositivos relevantes es de 50 Hz de alterna, y la tensión se encuentra sobre los 240V.

Finalmente, cabe destacar las gráficas de potencia y de porcentaje de carga de la batería. En la Figura 5.6 se aprecia como durante la noche cuando no hay luz solar la potencia en la batería es ligeramente positiva, descargándose como se puede ver en la Figura 5.7. Es en el momento en el que comienza a incidir el sol, cuando empieza a cargarse la batería al tener una potencia negativa que llega a un máximo de alrededor de unos -3kW y se alcance un 92 % máximo de batería. A partir de ese instante, la batería se mantendrá en ese porcentaje. Cuando caiga el sol será cuando comience de nuevo a descargarse la batería y volverá a haber potencia positiva en ella.

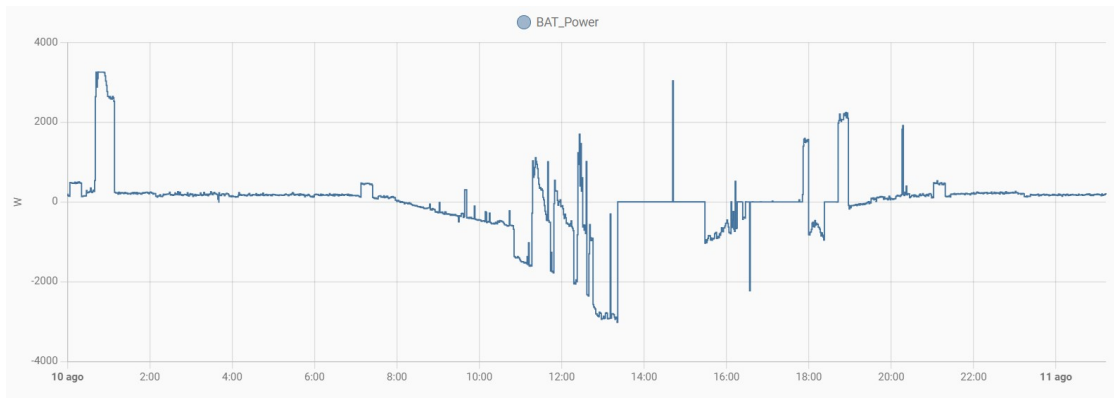


Figura 5.6: Potencia de la batería (W)

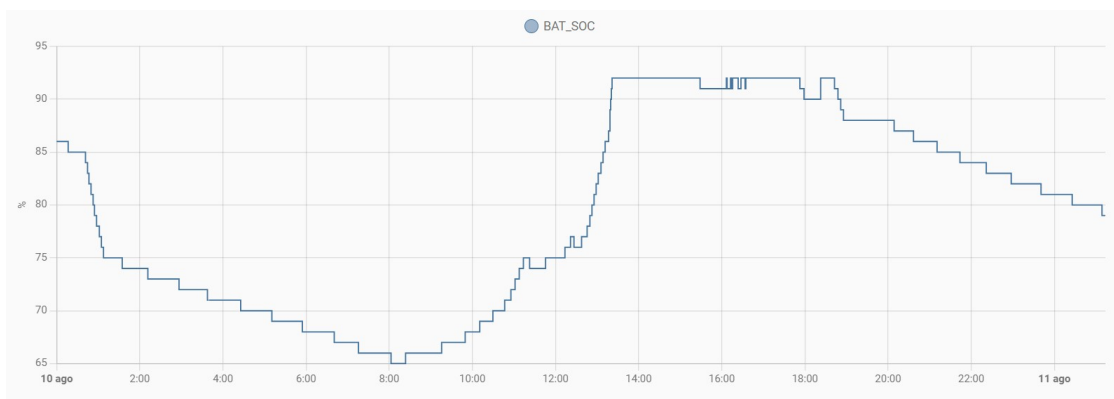


Figura 5.7: Porcentaje de carga de la batería (%)

Capítulo 6

Conclusiones y líneas futuras

6.1. Introducción

En este capítulo se comentan las conclusiones extraídas del trabajo realizado y tras el análisis de los resultados obtenidos.

6.2. Conclusiones

En este trabajo de fin de grado se ha diseñado y desarrollado un sistema que basado en el software Home Assistant permite gestionar, controlar y automatizar de manera domótica un entorno. Tras los resultados obtenidos se puede concluir que el uso de la energías renovables son buena solución para presente y futuro. Es cierto que en época de mayor calor y mayor número de horas de sol, como es verano, afecta a la energía generada al día en un panel fotovoltaico, aportando menor energía en invierno al disponer de menor incisión de luz solar sobre los paneles. No obstante, la energía recibida de los paneles fotovoltaicos es muy buena solución para complementar a la energía obtenida de la red eléctrica u otras fuentes como, por ejemplo, una posible entrada de gas natural.

Realizando comparación entre ambas energías, en el momento que la solar supera la obtenida por la red eléctrica, puede ser utilizada como única fuente para alimentar los distintos dispositivos del entorno, dándole siempre prioridad a este tipo de energía en el caso de poder usarla de manera eficiente y siempre que esta sea suficiente para su uso en cualquier dispositivo sin problemas. Por otro lado, cuando la energía solar captada sea inferior a la que se puede lograr a la de la red eléctrica será esta última la que prevalezca y alimente al conjunto del entorno y sus dispositivos. No obstante, la energía solar que sobra se podrá tanto usar para alimentar las baterías en caso de ser necesario como de devolver la energía a la propia red eléctrica.

Respecto a los protocolos de comunicación utilizados se puede llegar a la conclusión que la gran mayoría de dispositivos y sistemas energéticos a controlar en un entorno como puede ser una edificación, vivienda o industria se basan en protocolo Modbus. Se trata de un protocolo muy flexible, altamente utilizado tanto si es tipo RTU como TCP, siendo este último muy útil para acceder a la información de los distintos dispositivos. A partir de una dirección IP, un puerto y una serie de registros se consigue el control del equipo obteniendo sus datos de manera rápida y sencilla. A su vez, el uso de protocolos

en un ambiente IoT como son Zigbee, Bluetooth, WiFi o MQTT permite de manera muy eficaz el control y gestión de un entorno domótico, consiguiendo como se ha visto enlazar distintos sensores o elementos que permitan obtener información del estado del entorno, enviar/recibir mensajes o incluso activar ciertas salidas.

Home Assistant es un software para gestión domótica muy potente que permite integrar gran cantidad de dispositivos o equipos, como distintos tipos de comunicaciones permitiendo controlar todo un sistema amplio en una única interfaz. Tras su uso, se concluye que la programación mediante Python de Home Assistant es la indicada ya que utiliza menos recursos que el propio sistema de automatización que ofrece el software, permitiendo un amplio abanico de posibilidades. A su vez, Home Assistant es un software con una interfaz muy intuitiva y altamente personalizable consiguiendo que la adaptación a un entorno nuevo como es el de control domótico mediante un software dedicado como es el que se ha introducido sea muy agradable.

6.3. Líneas futuras

El objeto de este trabajo da por concluido el alcance del proyecto. Como posibles consideraciones a futuro, se propone:

- A partir del procesador i.MX93 del SOM utilizar su NPU para desarrollar algoritmos que permitan mediante machine learning o deep learning aprender costumbres de las personas y aplicarlos en un software domótico como Home Assistant.
- Continuación con la gestión energética de un entorno analizando energías distintas a la eléctrica y solar como puede ser gas natural.
- Aplicación del software domótico Home Assistant en entornos como puede ser en agricultura para ayudar en la gestión del sistema de riego, estado del campo, estado de invernaderos o supervisar condiciones para el cultivo de plantas.
- Integración de sensores meteorológicos en Home Assistant para la creación de una estación meteorológica personalizada y obtener datos en tiempo real sobre el estado climático local.

Apéndice A

Anexos

A.1. Anexo 1: Análisis detallado de las alternativas de software

Para poder elegir un software adecuado a las características que se buscan en este proyecto, se realiza un análisis de distintos entornos observando una serie de aspectos que permitan suplir las necesidades que acontecen en este trabajo.

Por ello en primer lugar se van a analizar los objetivos y características a cubrir.

A.1.1. Objetivos y características del software

Entre los requisitos que se buscan para la elección de un entorno adecuado, se desea que este cumpla las siguientes cualidades:

- En primer lugar, el software a utilizar debe permitir el uso de protocolos de comunicación tanto industriales (MODBUS, CAN o ETHERNET), como protocolos domésticos (Bluetooth/BLE, WiFi o Zigbee) entre otros.
- A su vez, debe existir la posibilidad de enviar/recibir datos e información de la nube, o cualquier plataforma cuyo objetivo sea similar. Del mismo modo, será necesario que haya viabilidad de funcionamiento en modo offline, en caso de requerir de acceso al entorno sin conexión a una red de internet.

Por otra parte, será esencial la capacidad de acceder de manera remota al software, característica que será necesaria si se quiere entrar al entorno o realizar cualquier acción sobre este fuera del dispositivo principal, o fuera de la vivienda o lugar en el que se ha automatizado y se va a estudiar.

- Finalmente, será recomendable que este entorno disponga de una interfaz web que posibilite una clara visualización, facilidad de trabajo y accesibilidad al control de los recursos. De la misma manera, será ventajoso que el software a escoger sea de código abierto (Open Source), para poder acceder de manera gratuita o lo más libre posible.

La disponibilidad de otras características que hagan interesante y atractivo al software serán relevantes, como la capacidad de realizar autodiagnósticos para

encontrar problemas o errores en su propio funcionamiento; el lenguaje de programación en el que se tendrá en cuenta la experiencia en su uso y sencillez, o el sistema operativo sobre el que se pueda instalar el software.

Por tanto, para realizar este análisis se han escogido los siguientes softwares: OpenHAB, Domoticz, RexControls, Codesys, OpenRemote, EdgeXFoundry, ROS, Home Assistant, Microsoft Azure IoT Edge, Node-RED y Homebridge.

A.1.2. OpenHAB

Open Home Automation Bus, es un software tecnológico y de código abierto para la automatización de la casa, funcionando como núcleo de una casa inteligente.



Figura A.1: Logotipo de OpenHAB

OpenHAB dispone de distintas fortalezas, entre las que destacan: Su capacidad de integrar multitud de dispositivos y sistemas en una única solución. Provee al mismo tiempo, de una interfaz de usuario uniforme y una variedad de reglas de automatización a lo largo de todo el sistema, sin tener en cuenta el número de fabricantes y subsistemas envueltos. Además de ser una herramienta flexible, permitiendo realizar gran variedad de automatizaciones. [56]

Con una arquitectura conectable, openHAB permite más de 200 tecnologías, dispositivos y sistemas diferentes. A su vez, es un software muy flexible, pudiendo ser ejecutado en diferentes sistemas operativos como pueden ser Linux, macOS, Windows, Raspberry Pi... Siendo capaz de la misma forma de acceder a él con aplicaciones para la web, iOS, Android y otros.

OpenHAB se ejecuta en su hardware sin necesidad de requerir de ningún servicio en la nube para funcionar, manteniendo sus datos privados y comunicándose directamente con sus dispositivos locales si hay posibilidad. No obstante, es a su vez compatible con la nube permitiendo integraciones basadas en la nube, permitiendo OpenHAB Cloud gratuito o alojando en uno propio.

Respecto a los lenguajes de programación, se dispone en primer lugar de un sistema de bloques, “*blockly*”, basado en una programación gráfica y sencilla con bloques. Por otro lado, habilita un lenguaje creado específicamente para openHAB basado en Xtend, llamado “*Reglas*”, y que se usa para procesos de automatización que invocan scripts funcionando como tareas. Finalmente, una versión antigua de JavaScript que funciona como editor de texto del lenguaje *blockly* previamente mencionado, sin embargo, no es recomendado su uso. No obstante, se pueden instalar otros diferentes lenguajes como añadidos, tan diversos como Python, Ruby, Groovy o Java entre otros.

En relación a los protocolos de comunicación, se posee tanto de industriales como domésticos, encontrando de los primeros, Modbus actuando tanto como cliente TCP, adquiriendo datos de los servidores Modbus TCP (master-slave), o funcionando como Serial master, tomando datos de Modbus serial slaves. A su vez, también hay otros protocolos disponibles en distintos aspectos como pueden ser bus CAN o Ethernet.

En cuanto a los protocolos domésticos, hay disponibilidad de Bluetooth/BLE, Zigbee y posibilidad de comunicación vía WiFi. Habrá también accesibilidad a MQTT, sin embargo, no se provee de un servidor dentro de este *binding* sino que se podrá usar libremente cualquiera disponible entre los Brokers MQTT, como “*Mosquitto*”.

Finalmente, se dispone de modo offline, permitiendo enviar órdenes y comandos sin estar en línea. Además, OpenHAB admite la comunicación de manera remota con sus servidores de manera bidireccional, permitiendo comunicarse con el servidor en distintos dispositivos y lugares, con varios sistemas como el uso de una VPN, servicios nube o ejecutando a través de una Reverse Proxy. En cuanto a la interfaz de usuario, cuenta con una app móvil para accesos con dispositivos móviles y una interfaz web con la que poder manejar y estructurar el software. Adicionalmente, hay posibilidad de realizar autodiagnóstico para encontrar posibles fallos y errores en el propio sistema.

A.1.3. Domoticz

Sistema de Automatización que permite monitorizar y configurar distintos dispositivos, sensores y medidores de temperatura, lluvia, viento, UV, agua, gas... Siendo capaz de enviar alertas y notificaciones a cualquier dispositivo móvil. Es de código abierto y programable en una variedad de sistemas operativos como pueden ser Linux, Windows, macOS o dispositivos embebidos en los que instalarlo. Con interfaz de usuario web HTML5, escalable y adaptada tanto para escritorio como para dispositivos móviles.[57]

Entre los lenguajes de programación utilizables en este software destaca por su sencillez *Blockly*, en el que como ocurría en OpenHAB, la programación está basada en un sistema de bloques. Del mismo modo, se dispone de opciones en Python, LUA o dzVents.

Respecto al acceso al entorno, Domoticz posibilita la entrada desde un punto remoto de manera totalmente segura a través de métodos como, VPN o servidor web Proxy entre otros, lo que permitiría a su vez la realización de conexión con la nube. Por otro lado, en el caso de carecer de una conexión activa con internet, Domoticz no será dependiente de ella tras ser instalado, siendo por tanto posible el acceso y trabajo de manera offline. Como característica adicional, de la misma manera que ocurría en OpenHAB, se podrá realizar autodiagnóstico del propio software.



Figura A.2: Logotipo de Domoticz

Por último, teniendo en cuenta los protocolos e integraciones disponibles en este software, se posibilita la accesibilidad a distintos tipos de comunicaciones industriales, como pueden ser ethernet con protocolos TCP/UDP IP y HTTP, comunicación Modbus o bus CAN entre otros, y protocolos domésticos como son Bluetooth, Zigbee, WiFi u otros como MQTT al acceder con Domoticz a un Broker que se enlace con los distintos dispositivos como puede ser *Mosquitto*.

A.1.4. RexControls

RexControls es una empresa orientada a la automatización e IoT, que tiene como objetivo proporcionar herramientas y servicios para desarrollar aplicaciones de control avanzado integrando las novedades del control automático, robótica, TIC, IA...[58]

Han desarrollado un sistema de control denominado REXYGEN, que permite desarrollar soluciones de distintos niveles en automatización, siendo un software seguro, innovador y de fácil acceso. Por tanto, REXYGEN es un software de control en tiempo real personalizable, ofreciendo una programación gráfica con distintos tipos de bloques y diagramas.



Figura A.3: Logotipo de RexControls

Ofrece una interfaz hombre-máquina (HMI) basada en web, un entorno de desarrollo unificado para los procesos a ejecutar tanto en Linux como en Windows, siendo estos los sistemas operativos disponibles o ejecutandolo en plataformas como Raspberry Pi o PLCs. El uso de este software será de extrema sencillez para personas que estén acostumbradas a utilizar Matlab-Simulink.

Por un lado, cabe destacar su lenguaje de programación basado en el desarrollo en control y automatización integrando bloques de funciones, que permiten que el proceso de programación sea rápido, sencillo y ameno para el programador.

Por otro lado, este software implementa una interfaz hombre-máquina simple, incorporando un servidor web para incluir esta interfaz, convirtiendo al dispositivo en una solución todo en uno. Esta interfaz para escritorio, Tablet o teléfono inteligente se puede generar automáticamente o diseñarse manualmente, basándose en HTML5 y funcionando con los navegadores comunes en Windows, macOS, Linux, iOS y Android.

REXYGEN permite el uso de gran variedad de protocolos de comunicación, tanto industriales como pueden ser Modbus TCP, Ethernet, CAN bus y CANopen; como domésticos como Bluetooth, WiFi, Zigbee, MQTT y otros de manera nativa, proporcionando comunicación con PLCs de Omron y Siemens.

Finalmente, REXYGEN da gran variedad de posibilidades, permitiendo comunicación con la nube, no obstante, si no se desea la conexión, el uso puede ser completamente independiente a esta función. A su vez, permite el acceso al software de manera offline permitiendo realizar ciertas características sin necesidad de conexión a la red, y un posible acceso de manera remota en caso de necesitar acceder desde distintos puntos. Independientemente de la complejidad del algoritmo también dispondrá de diagnósticos detallados pudiendo usar datos en tiempo real o archivados para ajustar los proyectos.

RexControls sin embargo dispone de ciertas limitaciones, necesitando comunicarse con el fabricante para comprobar compatibilidades, a pesar de una buena variedad de dispositivos afines. Por último, y como mayor desventaja, cabe indicar que no es de código abierto, teniendo que pagar para poder hacer uso de sus características, siendo por tanto un software de pago.

A.1.5. Codesys

Plataforma de software para tecnología de automatización industrial, centrándose en la herramienta de programación IEC-61131-3 (Sistema de desarrollo CODESYS). Codesys ofrece soluciones integradas y orientadas a la práctica para la configuración conveniente de aplicaciones de automatización. Se trata de un software de código abierto pero limitado únicamente al S.O. Windows, con funciones de seguridad y una conexión conveniente a la plataforma de administración basada en la nube, pudiendo realizar intercambio de datos entre cualquier red IoT fácilmente.[59]

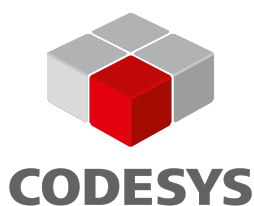


Figura A.4: Logotipo de Codesys

Codesys incluye una variedad extensa de funciones para dispositivos de automatización, habilitando comunicaciones tanto industriales (CANopen, CAN bus, Ethernet, PROFINET, Modbus), como domésticas (Bluetooth, Zigbee, Wi-Fi) o MQTT mediante distintas librerías.

Dispone para su uso distintos tipos de lenguajes de programación de sistemas de automatización y PLCs entre los que se pueden encontrar Structured Text (ST), Ladder Diagram (LD), Function Block Diagram (FBD) o Sequential Function Chart (SFC), pudiendo realizar su instalación en diferentes sistemas operativos como son Linux, Windows, macOS o sistemas embebidos como en una Raspberry PI. Sin embargo, solo se encuentra de manera gratuita para Windows, por lo que para cualquier otro sistema este software es de pago. No obstante, su versión más básica es gratuita.

Por otro lado, posee una interfaz de usuario única de manera completa y con un sistema de programación que abarca desde el previamente nombrado de PLCs clásico hasta una programación de controladores orientada a objetos, con buena integración de componentes adicionales para ingeniería de proyectos, variedad amplia de add-ons y fácil conexión a una plataforma “Industry 4.0” para administrar entornos de control por la nube. Por lo que el trabajo se realizará de manera local en la máquina teniendo el nivel de administración en la nube.

A su vez, será posible un teleacceso al dispositivo a través de Ethernet o bus de campo sin requerir software adicional, por lo que se dispone de acceso remoto al dispositivo. De la misma manera el software se podrá utilizar tanto de manera offline, como en línea.

Por otro lado, posibilita un seguimiento/diagnóstico del software a través de tecnologías de Internet, mediante distintas interfaces y elementos propios.

Finalmente cabe destacar que como se ha mencionado es un software esencialmente de pago, pudiendo encontrar otros similares de código abierto y mayores facilidades.

A.1.6. OpenRemote

Plataforma de administración de dispositivos IoT de código abierto (Open Source), totalmente intuitiva y fácil de usar, que incluye gran variedad de funciones y

características, desde la conexión de dispositivos hasta la creación de aplicaciones y aplicaciones inteligentes específicas.[60]

Incluye una interfaz de usuario web para crear conectar y administrar activos personalizables y almacenables, permitiendo a su vez el uso de servicios de localización con resúmenes geográficos y visualización de datos.

Por otro lado, OpenRemote permite acceso multiusuario y de su misma manera acceso remoto, con accesos principalmente a sus recursos propios desde distintos lugares, pero a su vez pudiendo conectarse mediante distintos roles limitados, de forma externa.



Figura A.5: Logotipo de OpenRemote

OpenRemote puede funcionar tanto como servidor como cliente según las necesidades conectando cualquier dispositivo, API y puerta de enlace. A su vez, ofrece distintos protocolos de comunicación para conectar dispositivos y servicios, incluyendo entre ellos HTTP, MQTT, Bluetooth, WiFi, Zigbee, Serial, u otros como TCP, Modbus, CAN bus o Ethernet.

Dispone de una API de administrador que permite que dispositivos o servicios externos se conecten como clientes, incluyendo una API MQTT (Broker), HTTP y Websocket.

En cuanto a la programación, se podrá automatizar sin necesidad de escribir una palabra de código, mediante *reglas When-Then* se puede monitorear cualquier atributo o condición y desencadenar una acción. De otra forma, mediante *reglas de flujo* se pueden procesar y manipular datos enlazando cadenas de procedimientos de manera gráfica. De la misma manera, se puede programar mediante Groovy que es un lenguaje de secuencia de comandos que permite definir lógicas de automatización más complejas, pudiendo definir casi cualquier cosa con este método.

A su vez se permite una aplicación móvil que facilita la instalación, monitoreo y mantenimiento, brindando opciones de mensajería y control basadas en la ubicación.

Entre otras características accesibles por OpenRemote, destacan la posibilidad de acceso a la nube, como también el acceso al software de manera tanto online como offline.

Por último, este software permite realizar diagnósticos y controlar posibles fallos para un uso seguro y fiable. Su instalación será viable en distintos sistemas operativos tanto de Linux, como en Windows 10 y versiones posteriores; también podrá ser instalado en una SBC, como por ejemplo una Raspberry PI.

A.1.7. EdgeXFoundry

EdgeXFoundry es una plataforma de código abierto (Open Source), altamente flexible y escalable que facilita la interoperabilidad entre dispositivos y aplicaciones en el edge de IoT.

Puede funcionar almacenando y usando datos tanto en la nube como de manera local, recopilando información de sensores, enviandola y recibiendo mediante el software. También permite el procesamiento de datos perimetrales y realización de IA (Inteligencia Artificial) en el alrededor.

EdgeXFoundry es independiente del hardware, sistema operativo, software y la nube, pudiendo trabajar sobre este en el software en distintos S.O. como pueden ser Linux, Windows, macOS o en cualquier SBC que sea necesario su uso como puede ser una Raspberry PI. Del mismo modo admite el método de acceso a la nube que el usuario prefiera.[61]



Figura A.6: Logotipo de EdgeXFoundry

Por otro lado, este software permite el uso de distintos servicios de dispositivos y protocolos de comunicación entre los que se pueden encontrar Modbus TCP, Ethernet, CAN Bus, Bluetooth para BLE, Zigbee, WiFi o MQTT entre otros. Respecto a este último, la configuración de conexión de MQTT Broker se consolida en un solo cliente MQTT compatible con SecretStore para las credenciales de autenticación, EdgeX se comunicará con el broker *mosquitto* y este lo hará con el dispositivo MQTT correspondiente, publicando mensajes en ambos sentidos y suscribiendo para recibir mensajes de EdgeXFoundry a *mosquitto* y, por otro lado, del dispositivo MQTT a *mosquitto*.

En cuanto a los lenguajes de programación a utilizar en este software se encuentran principalmente Go Lang y C, con los que se cubre todo el ámbito de programación, usando uno u otro según las necesidades.

Respecto a la interfaz de usuario, se trata de una interfaz intuitiva y amigable que permitirá una comunicación humano máquina sencilla.

Finalmente, EdgeXFoundry posee un servicio de dispositivo que permite una personalización, configuración sencilla y un autodiagnóstico de posibles problemas y así hallar una solución de la manera más rápida posible. Del mismo modo, en el caso de querer acceder de manera remota al software esto será posible mediante conexión cifrada a partir de SSH; con un SSH-Tunneling se permitirá navegar por el software de manera segura estableciendo un túnel contra un servidor remoto. Por último, si se deseara acceder de manera offline, esto sería posible, pudiendo realizar ciertas tareas sin estar en línea.

A.1.8. ROS

El sistema operativo de robot (ROS), es un software de código abierto compuesto por un conjunto de bibliotecas de software y herramientas que lo ayudan a crear aplicaciones de robot. Se trata de un software muy conocido en el sector de la robótica y con vasto ecosistema de software al tener una comunidad de desarrolladores y usuarios muy amplia, ya que es un software desarrollado por y para esa comunidad y quienes serán sus administradores en el futuro. A su vez es un software flexible y da libertad a personalizarlo según las necesidades.[62]



Figura A.7: Logotipo de ROS

Del mismo modo, ROS es multidominio no solo limitándose a la robótica, sino que abarca desde uso en interiores hasta exteriores, como del hogar a la automoción o del consumidor a la industria, pudiendo usarse por tanto para tareas de desarrollo IoT.

Es a su vez multiplataforma pudiéndose utilizar y siendo compatible con Linux, Windows y macOS, así como en otras distintas plataformas integradas (con micro-ROS), lo que permite un desarrollo e implementación perfectos de la autonomía en el robot, gestión del back-end y las interfaces de usuario.

Por otro lado, respecto a la programación de este software, podrá realizarse en distintos lenguajes probados por ROS como son Python, C++ y Lisp, aunque también se puede acceder a bibliotecas experimentales en Java y Lua.

En cuanto a los protocolos de comunicación disponibles en ROS, se encuentran una gran variedad entre las que destacan industriales como Modbus TCP, CAN bus, Ethernet, u otros domésticos como pueden ser BLE/Bluetooth, Zigbee o WiFi entre otros. Como ocurría en softwares previos, se posibilita para aplicaciones IoT protocolo MQTT, funcionando como cliente MQTT pudiéndose comunicar con clientes que se ejecutan en dispositivos no basados en ROS, a través de un broker MQTT.

Si fuera necesario un acceso a ROS de manera remota, esto sería posible mediante TeamViewer o usando una conexión remota de línea de comandos como ssh, pudiéndose usar en distintos lugares y por distintas máquinas. Del mismo modo, si se demandara acceder de manera offline resultaría factible ya que se pueden instalar aplicaciones de uso offline y de esta forma usar ROS sin permanecer conectado a internet, siendo realizable la comunicación de los nodos de ROS mediante localhost.

Respecto a la interfaz web, se dispone de una interfaz que permite almacenar y visualizar los datos que ofrece una experiencia intuitiva para el usuario y para poder realizar distintos controles según lo que el usuario necesite.

Por último, y de manera adicional, ROS permite realizar autodiagnósticos y análisis del software para encontrar posibles errores y fallos o controlar que el proceso realizado es correcto, recolectando los datos de los drivers del hardware, analíticas y datos de registro, recopilando y publicando por tanto sus resultados.

A.1.9. Home Assistant

Home Assistant es un software de automatización del hogar de código abierto que prioriza el control local y la privacidad. Cuenta con una comunidad mundial muy extensa y es perfecto para ejecutarse en distintos sistemas operativos o dispositivos, con más de 1900 integraciones.[45]

Una vez integrados los dispositivos en el hogar se puede liberar el motor de automatización de Home Assistant con la capacidad de realizar gran cantidad de posibilidades.

Home Assistant mantiene los datos localmente pudiendo funcionar sin tener necesidad de acceder a una nube obligatoriamente. Su uso será mediante una interfaz web,

por tanto se manejará principalmente de manera online; no obstante, permite el uso de distintos dispositivos offline sin problemas, manteniendo el funcionamiento si el sistema no esta online. Como consecuencia, esto estará limitado ya que el resto de dispositivos que necesiten comunicación online no podrán funcionar correctamente durante ese tiempo.

El software se comunicará de manera local con los dispositivos y extraerá datos de la nube si no hay otra opción, procesando todo de manera local, esto no quiere decir que no haya acceso a la nube, todo lo contrario, Home Assistant permite el uso y análisis de datos almacenados o tomados de la nube de forma práctica.



Figura A.8: Logotipo de Home Assistant

A su vez, se podrán usar aplicaciones móviles complementarias para controlar rápidamente los dispositivos y recibir notificaciones cuando suceda algo relevante, o para enviar, por ejemplo, la ubicación de la casa si el objetivo es la detección de presencia, enviando los datos de manera directa al hogar sin acceso por parte de terceros.

Este software permite estar al tanto del uso de energía con su función de administración de energía en el hogar, pudiendo así optimizar producciones, planificar el uso de energía y ahorrar dinero.

Respecto a los sistemas operativos sobre los que se puede instalar este software son compatibles Home Assistant Yellow (propia del software), Raspberry PI, Windows, macOS, Linux, o sistemas genéricos x86-64 (Intel NUC, por ejemplo) entre otros.

Home Assistant permite programar automatizaciones, scripts y detalles sobre los dispositivos principalmente en YAML que es el principal lenguaje para configuraciones en este software. Sin embargo, Home Assistant está escrito en sí mismo en Python, que permite una amplia capacidad en creación de automatizaciones e integraciones usando este lenguaje, pudiendo por tanto usar también Python para programar lo que sea necesario.

En cuanto a sus integraciones, se trata de un software extenso con altas capacidades permitiendo un amplio abanico de protocolos de comunicación, tanto industriales como domésticos, como pueden ser Modbus de forma tanto serial, TCP o UDP, CAN bus, Ethernet, como del mismo modo WiFi, Bluetooth/BLE o Zigbee entre otros muchos. Respecto a MQTT, también se encuentra disponible; para configurarlo se elige un broker, que puede ser propio de forma privada, o recomendablemente usando un add-on de *mosquitto MQTT*, o un broker público.

Home Assistant contempla una interfaz web muy completa y de uso intuitivo para el usuario, ampliamente personalizable y con una gran cantidad de características e integraciones a las que puede el usuario acceder y controlar desde ella.

En el caso de que fuera necesario el acceso de manera remota al software esto sería viable, ya que se disponen de algunos métodos para ello, pudiendo, por ejemplo, usar

Remote UI tomando acceso desde la nube, esto es muy sencillo para usuarios que tienen el servicio de Home Assistant Cloud.

Finalmente, se dispone de una integración de diagnósticos que permite al usuario acceso a reportes del comportamiento del software y tener control de un correcto funcionamiento así pudiendo detectar posibles fallos.

A.1.10. Microsoft Azure IoT Edge

Azure IoT Edge es un tiempo de ejecución centrado en dispositivos, que le permite implementar, ejecutar y monitorear cargas de trabajo en contenedores.[63]

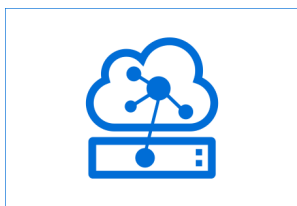


Figura A.9: Logotipo de Microsoft Azure IoT Edge

Se trata de una característica de Azure IoT Hub que permite escalar y administrar una solución de IoT desde la nube. Azure IoT Edge no solo limita a que todos los análisis se encuentren en la nube, sino que acerca el poder analítico de la nube a los dispositivos para generar mejores conocimientos y permitir tomar decisiones fuera de línea, por tanto, funcionando también de manera offline, pudiendo agregar datos localmente y luego solo enviar los conocimientos a la nube para su análisis, además en el caso de no haber conexión o tener conectividad intermitente, el software permitirá seguir usando los dispositivos de manera segura sincronizando con el estado más reciente cuando vuelvan a conectarse.

Por otro lado, Azure IoT Edge se compone de tres componentes, los módulos de IoT Edge que son contenedores que ejecutan servicios de Azure, de terceros o su propio código y se implementan en dispositivos IoT Edge y se ejecutan de manera local en ellos. A su vez, se encuentra el tiempo de ejecución de IoT Edge, ejecutado en cada dispositivo IoT Edge y administra los módulos. Finalmente, una interfaz web basada en la nube que le permite monitorear y administrar de forma remota los dispositivos IoT Edge.

Respecto a su disponibilidad, Microsoft Azure IoT Edge está disponible en el nivel gratuito y estándar de IoT Hub, sin embargo, es solo para prueba y evaluación, en el caso de necesitar un desarrollo más extenso y uso de más características Azure IoT Edge es un software de pago, necesitando realizar un cierto pago para disponer de las características necesarias.

Azure IoT Edge permite implementar procesamiento de eventos complejos, de aprendizaje automático, reconocimiento de imágenes y otras IA, teniendo servicios de Machine Learning, ejecutables localmente, sin estar limitados, pudiéndolos crear personalmente.

En cuanto a la programación, admite ejecutar el mismo código en un dispositivo o en la nube, siendo compatible tanto con Linux y Windows como sistemas operativos, u otros sistemas embebidos como puede ser una pequeña Raspberry PI (si no se procesan muchos datos), pudiéndolos escribir en distintos lenguajes como son Java, .NET Core 3.1, Node.js, C y Python, por lo que se dispone de variedad para adaptarse a distintas preferencias.

Por otro lado, mediante el tiempo de ejecución de IoT Edge se permiten distintas características interesantes entre las que destacan que IoT Edge informa sobre el estado del módulo a la nube realizando por tanto autodiagnósticos para saber cómo se encuentra el sistema en cada instante, y así poder realizar un monitoreo remoto, por lo que da accesibilidad a un uso del software de manera remota, que es importante.

Finalmente, cabe nombrar que Microsoft Azure IoT Edge permite el uso de una extensa variedad de protocolos de comunicación, entre los que se pueden encontrar Bluetooth, BLE, WiFi, NFC, Zigbee, LoRaWAN para conectar dispositivos móviles, seguros de área extensa de largo alcance u otras de uso industrial como pueden ser Ethernet, Modbus TCP o CAN bus. También se dispone de protocolo IoT MQTT, teniendo que usar uno de los dispositivos Azure IoT SDKs para construir los clientes de dispositivos IoT, a pesar de que en algunos escenarios será necesario usar librerías MQTT para comunicarse con tu IoT hub.

A.1.11. Node-RED

Node-RED es una herramienta de programación que conecta dispositivos hardware, API y servicios en línea de manera innovadora mediante código abierto. Proporcionando un editor basado en navegador que facilita la conexión de flujos mediante nodos ejecutables con un solo click.[64]

Se programará de manera sencilla mediante diagramas de bloques representados por nodos unidos unos con otros. A su vez las funciones que estarán escritas en JavaScript se pueden crear dentro del editor, usando un editor de texto enriquecido. Mediante una biblioteca integrada se permitirá guardar funciones, plantillas o flujos útiles para su reutilización.

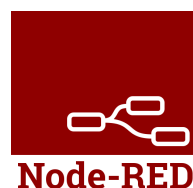


Figura A.10: Logotipo de Node-RED

Basado en Node.js el entorno de ejecución aprovecha al máximo su modelo sin bloqueo de eventos haciéndolo ideal para ejecutarse en el Edge, tanto en hardware de bajo coste como Raspberry Pi, como en la nube. Los flujos que se creen en Node-RED se almacenan mediante JSON fácilmente importable y exportable. Una biblioteca de flujos en línea permite compartir los mejores flujos con el resto del mundo.

Node-Red se podrá ejecutar libremente de forma local, en un dispositivo como Raspberry PI o interactuando con Arduino o Android, como de la misma manera en la nube mediante servicios de Amazon o Microsoft Azure.

En cuanto a los sistemas operativos compatibles con este software se encuentran Windows, Linux como Ubuntu o Debian, Raspberry PI, otros basados en RPM como Fedora o CentOS. De manera experimental se podría instalar en un sistema Android o mediante una máquina virtual en la nube a través de Microsoft Azure u otros.

Este software basado en Node.js apunta a un navegador/interfaz web para acceder al editor de flujo donde se podrá crear la aplicación mediante los nodos y conectándolos.

Respecto a los protocolos que permite utilizar Node-RED se encuentran distintos

industriales como son Modbus, CAN bus o Ethernet IP entre otros, y otros domésticos como BLE, Bluetooth, Zigbee o WiFi a parte de otros muchos. Respecto a MQTT, Node-RED permite conectarse a un Broker que se ejecute de manera local como puede ser *mosquitto*.

Por otro lado, en caso de necesitar acceso remoto al software será asequible mediante distintos métodos como la entrada a partir de la nube, VNC o una VPN.

Finalmente, este software podrá ser utilizado tanto en la nube como de manera local lo que permite un uso de manera offline, del mismo modo este software realizará informes de diagnóstico para conocer el estado del sistema en cada momento y posibles problemas.

A.1.12. Homebridge

Homebridge es un software de código abierto que integra dispositivos domésticos inteligentes que no son compatibles de forma nativa con HomeKit de Apple. Disponiendo de más de 2000 complementos que admiten miles de accesorios inteligentes diferentes.

Con Homebridge se podrán utilizar los distintos dispositivos de los que disponga el usuario de la marca Apple como pueden ser iPhone, iPad, Mac o Apple Watch para configurar automatizaciones, o accediendo a los accesorios a través de Siri.[65]



Figura A.11: Logotipo de Homebridge

Homebridge dispone de una interfaz web de usuario sencilla para administrar los complementos, configuración y accesorios, también permite instalar estos complementos y configurarlos, supervisar el servidor, y realizar copias de seguridad y distintos diagnósticos del sistema para poder controlar que el software se encuentre en estado correcto en todo momento.

En cuanto a la instalación de Homebridge, esta se puede llevar a cabo en distintos sistemas operativos entre los que se pueden encontrar macOS, Windows 10, Linux, Docker, otros dispositivos como pueden ser la Raspberry PI u otras plataformas.

Por otro lado, respecto a los lenguajes de programación predominantes en Homebridge son TypeScript y JavaScript, este último del que se desarrolla el entorno de ejecución sobre el que se basará el software que será Node.js.

Respecto al modo de ejecución del software, se podrá realizar de manera offline accediendo localmente al software, como también se podrán administrar datos y analizarlos a través de la nube, teniendo por tanto flexibilidad para su uso. De la misma forma, si fuera necesario el acceso al software de manera remota, se podrá realizar de distintas maneras como puede ser accediendo mediante ssh.

Homebridge dispone de distintos plugins que permiten la comunicación mediante tanto protocolos industriales como Modbus, CAN bus o Ethernet, u otros protocolos domésticos como son WiFi, BLE/Bluetooth o Zigbee entre otros. MQTT en Home-

bridge funcionará como un plugin dinámico que permite añadir y controlar accesorios con una API MQTT. Sin embargo, para que el enlace entre dispositivos hardware, APIs y servicios online se recomienda el uso de otro software adicional ya mencionado previamente como es Node-RED.

Finalmente, cabe recalcar que Homebridge está especialmente dedicado a dispositivos Apple y a su uso en estos, por lo que puede estar limitado en ciertos sentidos a la hora de querer acceder a dispositivos externos a la marca Apple.

Decisión del Software a utilizar

Software	Protocolos Industriales	Protocolos Domésticos	Accesibilidad	Interfaz de usuario	Open Source	Autodiagnóstico	Sistema operativo y lenguaje
OpenHAB	Disponible: Modbus, CAN bus, Ethernet	Disponible: Bluetooth/BLE, Zigbee, MQTT (uso de broker como Mosquitto), WiFi	-Comunicación con la nube: OpenHAB Cloud o servidor propio -Modo offline -Acceso remoto: VPN, servicios nube o Reverse Proxy	Interfaz web y aplicación móvil	Abierto	Disponible	- Sistema operativo: Linux, macOS, Windows, Raspberry Pi y otros como apps para web, iOS o Android. - Lenguaje: Blockly (gráfico), Reglas (lenguaje propio basado en Xtend), JavaScript u otros opcionales como Python, Ruby, Groovy o Java.
Domoticz	Disponible: Modbus, CAN bus, Ethernet con TCP/UDP, IP y HTTP	Disponible: Bluetooth, Zigbee, WiFi, MQTT (uso de broker como Mosquitto)	-Comunicación con la nube: servicios nube disponibles -Modo offline -Acceso remoto: VPN, servicios nube o servidor web Proxy	Interfaz web HTML5 escalable, para escritorio como para móviles	Abierto	Disponible	- Sistema operativo: Linux, Windows, macOS o dispositivos embebidos. - Lenguaje: Blockly (gráfico), Python, LUA o dzVents.
Rex Control System	Disponible: Modbus TCP, CAN bus, CANopen, Ethernet	Disponible: Bluetooth, Zigbee, WiFi, MQTT	-Comunicación con la nube: administración de datos en la nube -Modo offline -Acceso remoto disponible	Servidor web incorporado, para escritorio, tablet o teléfono inteligente basada en HTML5	Pago	Disponible	- Sistema operativo: Windows, macOS, Linux, iOS o Android. Para Raspberry Pi o PLCs también. - Lenguaje: lenguaje de programación basado en bloques de funciones (gráfico).
CODESYS	Disponible: CANopen, CAN bus, Ethernet, PROFINET, Modbus	Disponible: Bluetooth, Zigbee, WiFi, MQTT	-Comunicación con la nube: acceso a la nube posible no obligatorio -Modo tanto offline como en línea -Teleacceso a través de Ethernet o bus campo	Interfaz única con gran variedad de add-ons	Pago	Disponible	- Sistema operativo: Windows, macOS, Linux o sistemas embebidos como la Raspberry Pi. - Lenguaje: Structured Text (ST), Ladder Diagram (LD), Function Block Diagram (FBD) o Sequential Function Chart (SFC).
OpenRemote	Disponible: Serial, TCP, Modbus, Can bus, Ethernet	Disponible: Bluetooth, Zigbee, WiFi, HTTP, MQTT (broker)	-Comunicación con la nube: acceso disponible a la nube -Modo tanto offline como en línea -Acceso multiusuario y remoto disponible	Interfaz web y aplicación móvil para monitoreo y mantenimiento	Abierta	Disponible	- Sistema operativo: Linux, Windows 10 y posteriores versiones, también en SBCs como la Raspberry Pi. - Lenguaje: Programación mediante reglas When-Then, o mediante reglas de flujo de manera gráfica. También puede usarse Groovy.
Edgexfoundry	Disponible: Modbus TCP, Ethernet, Can bus	Disponible: Bluetooth/BLE, Zigbee, WiFi, MQTT (broker, p.ej. Mosquitto)	-Comunicación con la nube: funcionamiento posible tanto de manera local como en la nube -Modo offline -Acceso remoto mediante conexión cifrada con SSH	Interfaz web intuitiva y amigable	Abierta	Disponible	- Sistema operativo: Linux, Windows, macOS o en cualquier SBC como puede ser una Raspberry Pi. - Lenguaje: Programación en Go Lang y C (uno u otro según las necesidades).
ROS	Disponible: Modbus TCP, Ethernet, Can bus	Disponible: Bluetooth/BLE, Zigbee, WiFi, MQTT (cliente comunicando mediante un broker)	-Comunicación con la nube: posibilidad de uso -Apps instalables de uso offline -Acceso remoto mediante TeamViewer o con línea de comandos (ssh).	Interfaz web para almacenar y visualizar datos de forma intuitiva	Abierta	Disponible	- Sistema operativo: Linux, Windows, macOS o en plataformas integradas como micro-ROS. - Lenguaje: Programación en Python, C++ y Lisp, también hay alguna biblioteca en Java y Lua.
Home Assistant	Disponible: Modbus serial, TCP o UDP, CAN bus, Ethernet	Disponible: Bluetooth/BLE, Zigbee, WiFi, MQTT (con broker propio o add-on como Mosquitto)	-Comunicación con la nube: uso y análisis de datos de la nube -Funcionamiento offline con datos locales -Acceso remoto mediante Remote UI accediendo a la nube	Interfaz web completa, aplicaciones móviles adicionales	Abierta	Disponible	- Sistema operativo: Home Assistant Yellow (propio), Raspberry Pi, Windows, macOS, Linux o sistemas genéricos x86-64. - Lenguaje: Programación principalmente en YAML, sin embargo está escrito y permite su uso en Python.
Microsoft Azure IoT Edge	Disponible: Ethernet, Modbus TCP o CAN bus	Disponible: Bluetooth/BLE, Zigbee, WiFi, NFC, LoRaWAN, MQTT (uso de IoT SDKs para construir clientes de dispositivos IoT)	-Comunicación con la nube: análisis de los datos locales en la nube -Toma de decisiones fuera de línea (modo offline), datos locales -Administración remota de los dispositivos IoT Edge en la nube	Interfaz web basada en la nube	Pago	Disponible	- Sistema operativo: Linux, Windows y otros sistemas embebidos como Raspberry Pi. - Lenguaje: Programación en Java, .NET Core 3.1, Node.js, C y Python.
Nodered	Disponible: Modbus, CAN bus, Ethernet IP	Disponible: Bluetooth/BLE, Zigbee, WiFi, MQTT (conexión con un broker ejecutado localmente como Mosquitto)	-Comunicación con la nube: eventos ejecutables en la nube (Amazon o Azure) -Eventos también ejecutables de manera local (offline) -Acceso remoto mediante VNC o mediante una VPN	Interfaz web/navegador	Abierta	Disponible	- Sistema operativo: Linux y versiones como Ubuntu o Debian, Raspberry Pi, Fedora o CentOS, Windows o Android (experimental). - Lenguaje: Programación mediante diagramas de bloques unidos por nodos. Funciones escritas en JavaScript. Basado en Node.js.
Homebridge	Disponible: Modbus, CAN bus, Ethernet IP	Disponible: WiFi, BLE/Bluetooth, Zigbee, MQTT (plugin con una API MQTT)	-Comunicación con la nube: administración y análisis de datos en la nube -Modo offline accediendo de manera local -Acceso remoto con distintas maneras, como mediante ssh	Interfaz web disponible y uso de dispositivos Apple	Abierta	Disponible	- Sistema operativo: macOS, Windows 10, Linux, Docker, Raspberry Pi y otras plataformas. - Lenguaje: Programación en TypeScript y JavaScript, entorno basado en Node.js.

A.2. Anexo 2: Análisis detallado de las alternativas de hardware

Se desea encontrar un hardware óptimo para su aplicación en este trabajo y que permita realizar de forma rápida, adecuada y precisa los objetivos buscados en el proyecto cumpliendo todas las características necesarias. Por ello se analizan una serie de sistemas en módulos (System on Modules) y distintas soluciones propuestas por **Arrow Electronics**, que es un distribuidor tecnológico líder en innovación que asiste a fabricantes y proveedores; y a su vez ayuda al cliente a encontrar las soluciones adecuadas a sus necesidades.[66]

A partir de su guía **System on Modules and Solutions Guide**, se estudian distintas placas y sistemas embebidos, es decir, varios SOMs (System on Modules), SBCs (Single Board Computers) y soluciones integradas (Box Solutions) que minimizan el tiempo de diseño y riesgos.

Arrow muestra un amplio rango de placas y productos para microprocesadores y FPGAs de Intel, Microchip, NXP, Qualcomm, NVIDIA y ST Microelectronic basados en arquitecturas Arm (arquitectura RISC de 32 bits y, con la llegada de su versión V8-A, también de 64 Bits, desarrollada por ARM Holdings), y x86. Ofreciendo opciones con distintos grados de integración, se provee de un punto inicial ideal basado en requisitos técnicos y de negocio, incluyendo niveles de chip, placas (SOMs, SBCs) y soluciones en caja o integradas (Boxed Solutions).

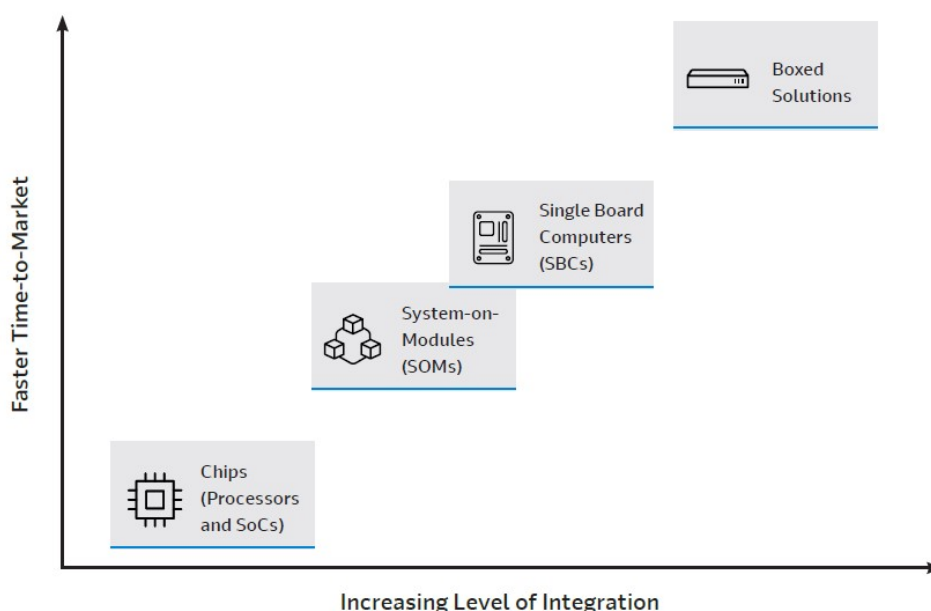


Figura A.12: Rapidez de comercialización respecto a su grado de integración

Como se observa en la Figura A.12 conforme más integrado se encuentra, más rápida es la comercialización del dispositivo, por tanto interesa encontrar un punto medio, pudiendo encontrar como las mejores opciones los SOMs y SBCs, por tanto estas serán las que se analizarán.

A.2.1. Arm

La revolución IoT ha ido en incremento durante los últimos años, teniendo un crecimiento importante en sensado, conectividad y edge processing.

Arm se encuentra en el frente de esta revolución con un 70 % de la cuota de mercado de dispositivos embebidos.

Sus ventajas con un ecosistema global y dinámico hacen que sea la elección de arquitectura en diversas vertientes IoT.

Por otro lado, un par de tendencias están emergiendo en el interior de la industria. La primera es que la industria está transitando a un software de modelo *‘nativo a la nube’*, en oposición al desarrollo de *nivel-dispositivo*. El incremento del número de terminales inteligentes y la masiva explotación de datos procesados por ellos lleva a este cambio.

La segunda es que hay una fragmentación en el ecosistema compuesto de distintos distribuidores de silicio, integradores de sistemas... Siendo la estandarización un reto.

Para todo ello, Arm comprende con una gran cantidad de socios en el ecosistema trabajando en colaboración, esforzándose para llevar un desarrollo cloud-native al edge y dispositivos terminales basándose en tres puntos, *Estándares, Ecosistemas y Seguridad*.



Figura A.13: Logotipo de Arm

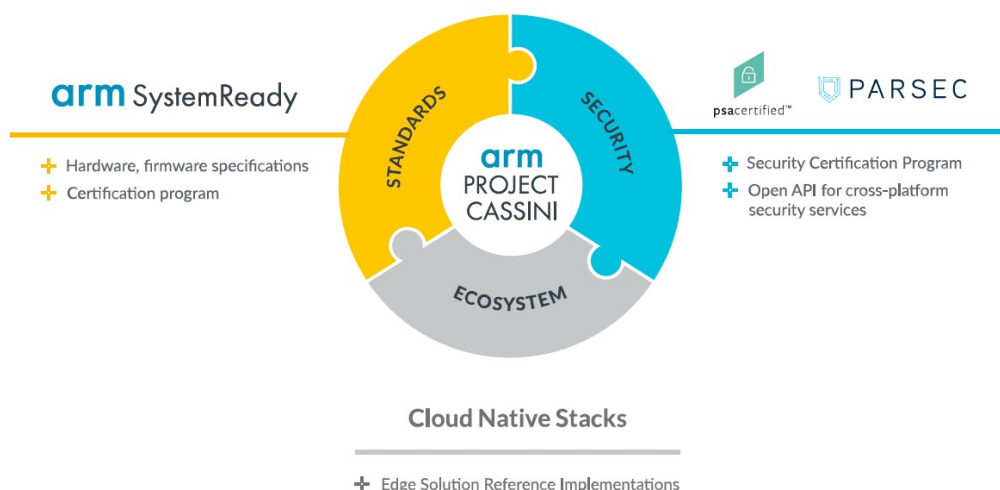


Figura A.14: Proyecto de Arm basado en los estándares, ecosistemas y seguridad

Finalmente, cabe nombrar que Arm dispone de una opción de *Hardware Virtual*, con el que intentan evitar el problema de pérdida de tiempo de los desarrolladores de software por encontrar disponibilidad de una placa y su configuración, sin tener que esperar a un hardware real, acelerando el time-to-market y cambiando las reglas de juego del ciclo de desarrollo de IoT.

A.2.2. Intel

En primer lugar se encuentra Intel, empresa dedicada a la fabricación de semiconductores y circuitos integrados, enormemente conocida por sus procesadores, y que en el rango de búsqueda solicitado para este proyecto dispone de una variedad de soluciones de arquitectura x86 y placas FPGAs, aunque estas últimas no se analizarán ya que no son parte del objetivo.



Figura A.15: Logotipo de Intel

A.2.2.1. Intel NUC

Se trata de un PC de pequeño formato que ocupa muy poco espacio. Exprime un poderoso procesador, con memoria rápida y almacenamiento masivo. Presentan los últimos procesadores Intel Core, disponible con graficas integradas o discretas, es sus configuraciones SSD o disco duro.

Los kits de barebones (cajas de PC de dimensiones reducidas) y placas son poderosos mini PCs para proyectos de todo tipo de escala, con aplicación en máquinas medicas como en aplicaciones industriales.

Entre sus beneficios destacan el tener un buen comportamiento, ser de tamaño pequeño, resistente a cambios de temperatura, flexible, eficiente energéticamente y personalizable.

Se dispone de Mini PCs, Kits o Laptops, sin embargo, para el caso a tratar no tiene sentido profundizar, además de:

- *Intel NUC Boards*: listas para integrar, con procesador soldado en ella e independiente del chasis para uso embebido y diseño personalizado.
- *Intel NUC Elements*: Familia de componentes modulares.



(a) Intel NUC Boards



(b) Intel NUC Elements

Figura A.16: Intel NUC Boards and Elements

- *Intel NUC 11 Compute Elements*: Compacto sistema en tarjeta que contiene procesador, memoria y suite de I/O en un sencillo paquete. Ofrecido con un completo rango de procesadores Intel.

Por otro lado, hay otras versiones con procesadores más potentes y con mejores características, pero están dedicados para aplicaciones que exijan un importante poder de procesamiento y mayor capacidad de la necesitada. Como pueden ser la variedad de versiones del Intel NUC 12 Extreme.

A.2.2.2. Asociados a Intel

- **EXOR**

Trata sistemas embebidos basados en arquitectura Arm, ofreciendo a los clientes la oportunidad de aprender a ser expertos en áreas de desarrollo de sistemas electrónicos con BSP, Linux, Yocto y Codesys, sistema de integración y conectividad a la nube. Incluye como servicios: Diseño Hardware, FPGA y desarrollo software, y aplicación a la nube.

Disponen de SOMs con arquitectura Arm, FPGA o x86, aunque tienen el problema de estar limitados a usar los Softwares/Firmwares disponibles (Linux, Yocto/Codesys). Ejemplo: *gS02 System on Module*.

- **IWave**

Dispone de un extenso portfolio de SOMs y SBCs para automoción, industria, medicina y aviación. Entre los servicios que ofrece destacan: Diseño de placa, software y hardware personalizable, manejo de su ciclo de vida, control de fabricación y producción, gran variedad de certificaciones. Por otro lado tendrá como softwares disponibles Linux, Quartus.

Ejemplos: *Stratix 10 SX/GX Series System On Modules*, *Arria 10 SX/GX Series System On Modules*.

- **Trenz Electronic**

Opera como proveedor de servicios de desarrollo para la industria electrónica, incluyendo tanto diseño en apoyo, como conjunto de diseños que cubren todos pasos de las especificaciones del producto. Tienen un prototipado rápido, experiencia en sistemas embebidos para ambientes industriales y largo ciclo de vida de los dispositivos. Poseen software disponible sin limitaciones.

Sus procesadores para SOM se basan en *Cyclone* de Intel.

A.2.3. Microchip

Microchip dispone de distintas opciones para FPGAs o de vía rápida a *Single-Pair Ethernet*, sin embargo, para la aplicación solicitada sería más interesante sistemas dedicados a IoT. Microchip para ello dispone de *Placas de desarrollo Shield96*.



Figura A.17: Logotipo de Microchip

Se trata de Hardware de seguridad avanzado para IoT a escala, basado en dispositivos Microchip y provee una solución aplicable a distintas vertientes IoT. Sigue soluciones basadas en Linux y lleva la seguridad de hardware por diseño.

Por lo tanto, aunque tiene opción dedicada a IoT, no es lo que se busca debido a que el objetivo a conseguir no se centra únicamente en seguridad.

A.2.4. NXP

NXP es una compañía global ampliamente conocida dedicada a la fabricación de semiconductores que crea soluciones de distintos tipos. Dentro de la amplia gama de dispositivos que posee, se van a analizar algunas versiones de SOMs y SBCs interesantes y bastante completas.



Figura A.18: Logotipo de NXP

A.2.4.1. i.MX 8 Series Application Processors

Las nuevas series de procesadores i.MX 8 de NXP se construyen en una tecnología con fiabilidad de una longevidad de más de 15 años, un software robusto y un amplio ecosistema, lo que apoya el hecho de continuar la tradición que tiene la empresa de ayudar a los clientes a encontrar el balance correcto entre prestaciones y poder/potencia; todo esto mientras ofrece soluciones escalables y seguras.

Estas series incluyen las familias i.MX 8M, i.MX 8X, y i.MX 8, compatibles con Android, Linux, QNX, Green Hills y otros sistemas operativos comerciales.

Entre las tecnologías que ofrece destacan, una avanzada seguridad, AI (Artificial Intelligence), Machine learning y visión, soluciones HMI, automatización industrial o infoentretenimiento.

Tiene como aplicaciones objetivo el control de construcciones, electrónica del consumidor, cuidado de la salud, dispositivos multimedia, video y audio o cuadro de mando de un automóvil entre otros.

A continuación se van a comentar las familias previamente nombradas:

- ***i.MX 8 Applications Processor Family*** (con Dual A72 Cores, Dual GPU)
Logrando un alto rendimiento hasta 1.6 GHz, la familia i.MX 8 soporta procesamiento multimedia y puede ejecutar algoritmos de procesamiento intensivo de Machine Learning. Con un dominio seguro y un innovador procesamiento de visión, revolucionando distintas aplicaciones de automoción, sistemas industriales, visión, HMI y SBCs. Basándose en arquitecturas Arm Core o DSP Core entre otras.
- ***i.MX 8X Applications Processor Family***
A disposición de una alta fiabilidad y seguridad, son eficientes y con un rendimiento acreditable. Están contruidos con un alto nivel de integración, soportando funciones de voz, audio, procesamiento de imágenes, video y graficas. Se trata de un procesador ideal para una seguridad certificable y una correcta realización de los requerimientos. Incluyendo las versiones *i.MX 8QuadXPlus*, *i.MX 8DualXPlus* y *i.MX 8DualX*.
- ***i.MX 8M Applications Processor Family***
Liderando en la industria en audio escalable, voz y procesamiento de video, combina alto rendimiento informático con capacidades líderes en el sector en audio, voz y video. Pueden llegar a alta definición HD de 4k en video streaming, calidad de audio profesional, reconocimiento de voz, AI, machine learning o edge computing. Usa tecnología avanzada presentando una gran eficiencia de potencia y un mejorado sistema de fiabilidad.

A.2.4.2. i.MX 9 Applications Processors

Basado en sistemas anteriores, esta serie de procesadores trae juntas unas altas prestaciones de los núcleos de aplicación, un dominio en tiempo real independiente basado en MCU, arquitectura Energy Flex, seguridad del estado del arte con enclave de seguridad EdgeLock, y un dedicado mecanismo de procesamiento de datos multi-sensorial. Dentro de este tipo de procesadores, NXP ha desarrollado la serie *i.MX 93*. Estos procesadores se basan y desarrollan unas tecnologías clave:

- *Eficiencia de energía.*

Optimizar el rendimiento se está convirtiendo en un requisito crítico para muchos diseños de dispositivos IoT. Se basa en Energy Flex de NXP que hace óptima la eficiencia en requerimientos críticos, a su vez un control de la potencia afinado se construye en estos procesadores para optimizar la energía ayudando a reducir la huella de carbono y alargando la vida de las baterías.

- *Consciente del edge.*

En el edge, los clientes necesitan una combinación balanceada de interfaces de sensores, procesamiento multimedia y aceleración de red neuronal en una óptima y equipada solución.

- *Seguridad.*

Integra una arquitectura avanzada que ofrece buena protección a los peligros y un actualizado protocolo de ayuda criptográfico.

A.2.4.3. Edge Computing

El procesamiento de SOC's contiene múltiples elementos de procesado, que pueden ser usados independientemente o juntos para realizar machine learning en el edge. Además, puede disponer de ISP para la mejora y uso de imagen y su procesado.

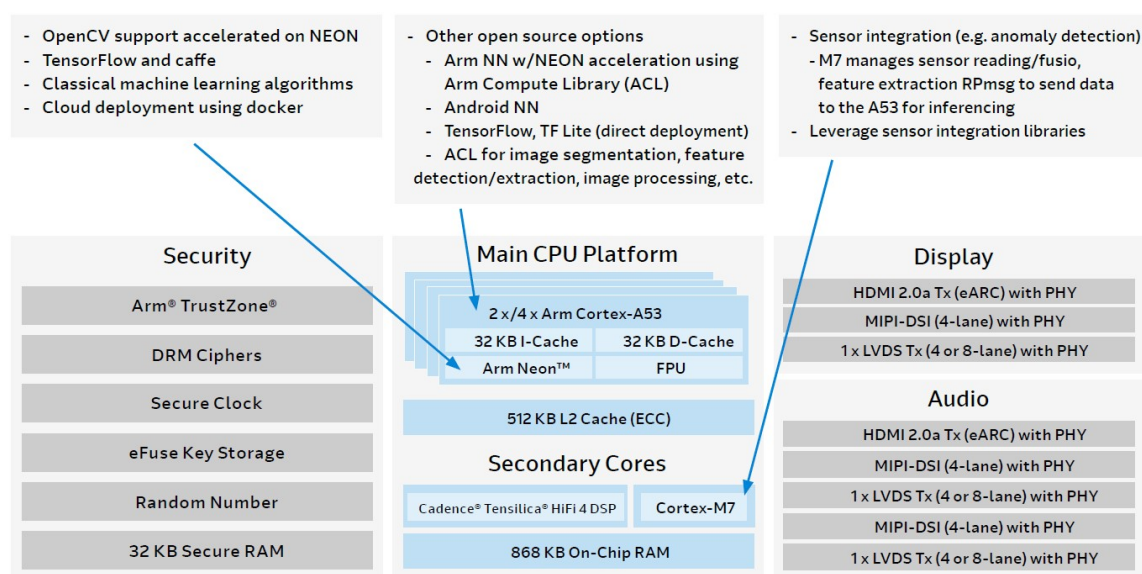


Figura A.19: Estructura de un procesador NXP (primera parte)

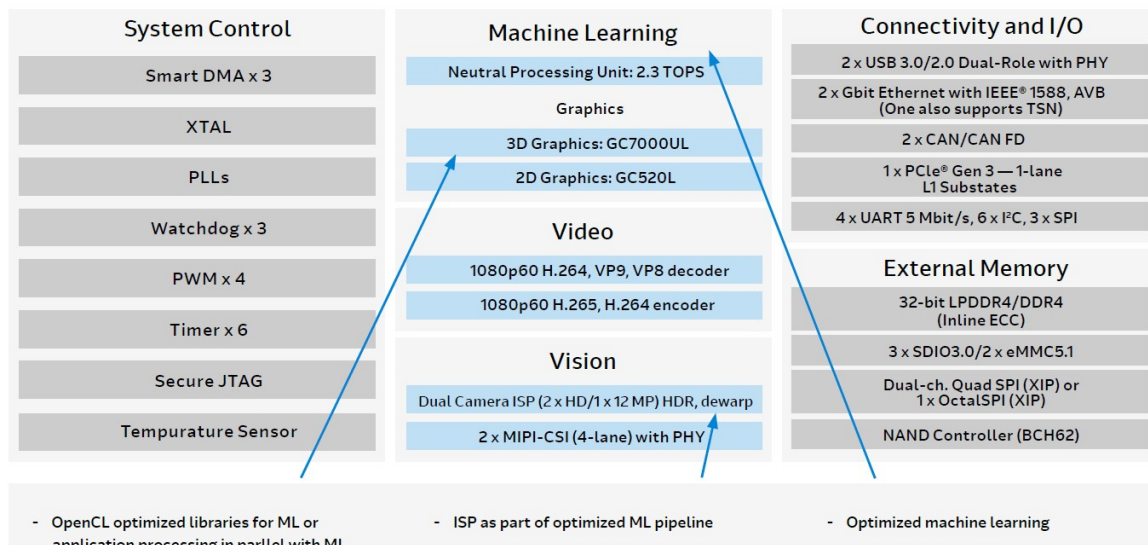


Figura A.20: Estructura de un procesador NXP (segunda parte)

El esquema representado de forma conjunta en ambas Figura A.19 y Figura A.20, muestra el esquema típico de un procesador NXP, en este caso con la capacidad de realizar Machine Learning de forma muy eficiente al disponer de una NPU (Neural Processing Unit). La capacidad de realizar procesamiento y aprendizaje automático no es un punto clave respecto a la toma de decisión del procesador por lo que no se le va a dar importancia. No obstante, el resto de su estructura es muy interesante, principalmente en la plataforma CPU como se puede apreciar en la Figura A.19. Se dispone de una CPU principal, y a su vez, puede haber un segundo *Core* con el que se podrían realizar, por ejemplo, procesos en *tiempo real*. Al mismo tiempo, tiene accesibilidad a soluciones de seguridad, displays, sistemas de control, audio, video, visión, memoria externa y distintos tipos de conectividad como Ethernet o CAN entre otros.

A.2.4.4. Asociados a NXP

■ **Boundary Devices**

Distribuidor de SOMs y SBCs basados en arquitectura Arm con procesadores NXP i.MX, encontrando SOMs de las series i.MX 8M e i.MX 9, y por otro lado SBCs con tan solo versiones i.MX 8M. Permitiendo instalar sobre ellos sistemas operativos Linux, Android, Debian, Yocto o FreeRTOS entre otros.

■ **DIGI**

Con más de 30 años de experiencia, se trata de un importante distribuidor de productos terminados, dispositivos integrados y software para aplicaciones IoT, centrándose en la calidad, seguridad y facilidad de integración además de la longevidad de sus productos. Se usará para ello sus familias de productos ConnectCore con SOMs y SBCs basados en procesadores NXP de series i.MX 8 e i.MX 6.

■ **iWave Systems**

Con gran variedad de SOMs y SBCs, disponen de productos con gran longevidad y revisando su correcto ciclo de vida. Ofrecen personalización de diseños de placas, software y hardware, control de procesos de fabricación y producción, y a su vez, distintos tipos de certificaciones (CE,FCC,RoHS...).

En cuanto a los procesadores, podrán contar con series i.MX 9, i.MX 8 y i.MX 6 de NXP, permitiendo el uso de varios softwares como Linux, Android o Ubuntu entre otros.

■ **Variscite**

Se trata de un proveedor miembro platino de NXP por lo que su prioridad con ellos es la más alta posible. Diseñan, crean, producen y con ello proveen un amplio abanico de SOMs basados en arquitectura ARM en el mercado de sistemas integrados con un gran rango de opciones de configuración.

En cuanto a sus fortalezas, disponen de una buena relación calidad/precio, con un servicio de ayuda estable y una longevidad de hardware y software mayor a 15 años.

Variscite ofrece *Cores* basados en las series de NXP i.MX encontrando versiones actuales como la i.MX 93 muy interesantes. Con posibilidad de conexión y características *Pin2Pin* y software a instalar disponible en Linux, Android, FreeRTOS o Yocto entre otros.

A.2.5. Qualcomm

Los procesadores de Qualcomm van más allá de la innovación en smartphones, haciendo del mismo modo las siguientes generaciones de dispositivos IoT más actualizados, conectados, inteligentes e interactivos, aportando por tanto en ámbitos diferentes como es en el caso de los dispositivos IoT.



Figura A.21: Logotipo de Qualcomm

Aportando con sus procesadores de IA distintas experiencias de usuario, como reconocimiento facial, soporte integrado de audio y cámara, conectividad integrada vía Bluetooth y WiFi haciendo a estos procesadores ideales para aplicaciones innovadoras. A su vez, se dispone de un alto rendimiento ante bajo uso de energía, teniendo una buena optimización de las plataformas.

Posee de distintos tipos de procesadores según su nivel de calidad: Premium Tier, High Tier, Mid Tier y Entry Level; sin embargo, aunque todos ellos tienen características más que suficientes para el objetivo, las de mayor calidad no serían muy rentables respecto a sus precios, teniendo versiones más económicas con funcionalidad suficiente.

Estos procesadores tienen como aplicaciones típicas, robots, conexión de cámaras, asistentes inteligentes, comercio o máquinas de venta entre otros.

Entre los procesadores más interesantes para la aplicación a desarrollar en este proyecto pueden destacar:

A.2.5.1. QCS610/QCS410 Application Processor

El primero de High Tier y el segundo de Mid Tier son SOCs entregados a un alto funcionamiento, eficientes energéticamente para las nuevas generaciones de cámaras y empresas inteligentes/home applications para un segmento intermedio. Con buena CPU, ancho de banda suficiente, una estupenda conectividad y variedad de características útiles para distintas aplicaciones. Incorporando al mismo tiempo notables

cualidades respecto a display, cámara, video y audio entre otros.
Por otro lado permiten la instalación exclusiva de Linux como sistema operativo.

A.2.5.2. Asociados a Qualcomm

- ***eInfochips***

Se trata de una compañía de *Arrow*, licenciada con procesadores de *Qualcomm Technologies* y ofrece SOMs, SBCs u otros productos para compañías para acelerar su salida al mercado. Dando servicio de hardware y firmware, AI/ML, acceso a la nube y otras aplicaciones similares.

Se basan en distintos tipos de procesadores Qualcomm, habiendo posibilidad tanto de la versión QCS610 como QCS410 favorables para este objetivo y con software disponible Android (aplicaciones móviles) o Linux.

- ***Penguin Solutions***

Fabricante y desarrollador de rápido crecimiento y con buen rendimiento, basados en arquitectura ARM para plataformas integradas con aplicación en IoT. Realizando avanzadas colaboraciones con Qualcomm, Penguin Solutions diseña y fabrica hardware inspirado en procesadores Snapdragon. Al mismo tiempo, ofrecen servicio de fabricación, hardware y software personalizados, ciclo de vida flexible en sus dispositivos y soporte técnico.

Ofrecen un abanico de SOMs y SBCs basados en distintos procesadores Qualcomm, con la virtud de poder tener como núcleo procesadores Snapdragon, muy bien valorados en el mercado de los smartphones. El software en el que se apoyen puede ser tanto Android como Linux.

Estudio de Alternativas Hardware

System on Modules (SoM)					
Referencia del Módulo	CPU	Memoria y almacenamiento	Conectividad	Sistema Operativo	Comentarios
gS01	-Nombre: Intel® Atom E39XX (XX=30, 40 ó 50) -Fabricante: Intel -Tipo: Intel Atom® Processor X Series -Nº de cores: 2/4 -Max CPU Clock: 2(1.8) -Real-Time Co-CPU: (Única CPU con 2 o 4 cores)	-RAM: hasta 8GB DDR4 -eMMC: hasta 64GB	-1 SD, 6 USB2.0, 2 USB3.0, 3 UART, 5 I2C, 1 SPI, 19 GPIO -CAN: 2 (Pero solo para la FPGA) -Ethernet: 1 Gigabit Ethernet -WiFi: - -Bluetooth: -	Soporta plataformas Windows y Linux, Linux BSP rt (OSADL) or Android BSP	Dispone de una FPGA de Intel Cyclone , por otro lado hay comunicación HDA y I2S para tratamiento de audio. En cuanto a su colocación en una posible carrier, el módulo iría soldado directamente a ella, con un tamaño adecuado requiere de alimentación de 5V y con una interfaz PCIe que mejora el rendimiento del módulo. Hay sin embargo problemas notables al solo disponer CAN para la FPGA y no poder realizar comunicación vía WiFi o Bluetooth .
Digi ConnectCore 93	-Nombre: NXP® i.MX 93 -Fabricante: NXP -Tipo: Cortex®-A55 (ARM) -Nº de cores: 2 -Max CPU Clock: 1.7 -Real-Time Co-CPU: Cortex-M33 a 250 MHz (ARM)	-RAM: hasta 2GB LPDDR4 -eMMC: hasta 32GB	-1 uSDHC, 2 USB2.0, 8 LPUART, 8 I2C, 2 I3C, 8 LPSP, 112 GPIOs, 1 ADC, 4 PWM -CAN: 2 CAN/CAN-FD (Flexible Data-Rate) -Ethernet: 2x 10/100/1000 Mbps -WiFi: Wi-Fi 6 inalámbrico 802.11ax de doble banda 1x1 -Bluetooth: Bluetooth® 5.2	Digi Embedded Yocto Linux	Con amplio ciclo de vida, tiene una NPU para análisis y desarrollo neuronal de aplicaciones AI/ML. Integra seguridad, identidad y privacidad del dispositivo Digi TrustFence (aspecto crítico para los dispositivos de Digi), y posible gestión remota con Servicios Nube de Digi ConnectCore. Dispone de 3 módulos de interfaz de audio síncrono (SAI). Gestión avanzada de energía, seguridad, soporte de periféricos y fiabilidad del sistema opcional con Digi Microcontroller Assist.
Digi ConnectCore 8M Nano	-Nombre: NXP® i.MX8 Nano -Fabricante: NXP -Tipo: Cortex®-A53 (ARM) -Nº de cores: 4 -Max CPU Clock: 1.4 -Real-Time Co-CPU: Cortex-M7 a 600 MHz (ARM)	-RAM: hasta 1GB LPDDR4 -eMMC: hasta 8GB	-3 uSDHC, 2 USB2.0, 4 UART, 4 I2C, 3 SPI, 112 GPIOs, 1 ADC, 10 canales PWM -CAN: - -Ethernet: 10/100/1000 Mbps -WiFi: Tecnología inalámbrica de doble banda 802.11a/b/g/n/ac -Bluetooth: Bluetooth® 5	Digi Embedded Yocto Linux	Contiene servicios de computación tanto en la nube como en el edge. Por otro lado la seguridad, identidad y privacidad las suple con Digi TrustFence, y dispone de gestión remota con servicios nube de Digi. En cuanto a gráficos contiene una unidad de procesamiento gráfico con 0.8GPx/s. Respecto a periféricos dispone de 5 módulos SAI para audio síncrono. Sin embargo, no soporta comunicación CAN bus que es necesaria. Hay posibilidad de asistencia al uC en el módulo mediante Digi Assist.
VAR-SOM-MX93	-Nombre: NXP® i.MX93 -Fabricante: NXP -Tipo: Cortex-A55 (ARM) -Nº de cores: 2 -Max CPU Clock: 1.7 -Real-Time Co-CPU: 250MHz Cortex-M33 (ARM)	-RAM: 512MB-2GB LPDDR4 -eMMC: 8-128GB	-1 SD, USB2.0, 4 UART, 5 I2C, 3 SPI, 1 JTAG, 4 ADC -CAN: 2 -Ethernet: 2x 10/100/1000 Mbps -WiFi: Certified single-band 802.11 b/g/n or dual-band 802.11 ac/a/b/g/n -Bluetooth: BT/BLE 5.2	Linux (Yocto, Debian), FreeRTOS	Variscite incluye Pin2pin Family la cual da ventajas notorias en cuanto a escalabilidad, relación calidad/precio, facilidad de desarrollo e integración y tiempo de vida; este módulo pertenece a esta familia. Por otro lado, hay disponible RTC pero en carrier. En cuanto a multimedia, se dispone de una interfaz de cámara MIPI-CSI2 y aceleración NPU para AI/ML . Diversas características para display como DSI o Touch controller y respecto al audio hay driver para auriculares, micro digital o analógico y distintas interfaces serie para audio digital. Wiki variscite muy útil.
VAR-SOM-MX8M-NANO	-Nombre: NXP® i.MX8M Nano -Fabricante: NXP -Tipo: Cortex™-A53 (ARM) -Nº de cores: 1 a 4 -Max CPU Clock: 1.5 -Real-Time Co-CPU: 650MHz Cortex™-M7 (ARM)	-RAM: 512MB-2GB DDR4 -SLC NAND: 256-512MB -eMMC: 8-128GB	-1 SD, USB2.0, 4 UART, 4 I2C, 3 SPI, 1 JTAG -CAN: 1 CAN/CAN-FD (Flexible Data-Rate) -Ethernet: 10/100/1000 Mbps -WiFi: Certified single-band 802.11 b/g/n or dual-band 802.11 ac/a/b/g/n -Bluetooth: BT/BLE 5.2	Linux (Yocto, Debian), FreeRTOS, Android	Pertenece a la familia Pin2Pin de variscite. Se puede disponer de RTC pero estará en carrier. En cuanto a multimedia se tiene una GPU Vivante G7000UL y una interfaz de cámara MIPI-CSI2 ; MIPI DSI, LVDS y touch controller para display. Respecto al audio hay driver para auriculares, microfono digital y analógico, y distintas interfaces serie para audio digital. Wiki variscite muy útil.
VAR-SOM-MX8	-Nombre: NXP® i.MX 8QuadMax / 8QuadPlus -Fabricante: NXP -Tipo: 2x Cortex™-A72 + 4x Cortex™-A53 (ARM) -Nº de cores: 5 a 6 -Max CPU Clock: 1.6 GHz Cortex™-A72, 1.2 GHz Cortex™-A53 -Real-Time Co-CPU: 2x 266MHz Cortex™-M4F (ARM)	-RAM: 2-8GB LPDDR4 -eMMC: 8-128GB	-1 SD, USB3.0/2.0, 5 UART, 4 I2C, 4 SPI, 1 JTAG, 1 keypad (5x4 matrix), PCI-Express 3.0 -CAN: 2 CAN/CAN-FD (Flexible Data-Rate) -Ethernet: 2x 10/100/1000 Mbps -WiFi: Certified 802.11 ac/a/b/g/n -Bluetooth: BT/BLE 5.2	Linux (Yocto, Debian), FreeRTOS, Android	Pertenece a la familia Pin2Pin de variscite. Procesador HiFi4 DSP para procesamiento de señales digitales. En cuanto a multimedia, habrá 2 GPU Vivante GC7000VX e interfaz cámara MIPI-CSI2 ; se dispone de HDMI , LVDS, DSI y touch controller entre otros para Display, y respecto a audio, hay driver para auriculares, microfono digital o analógico e interfaz serie para audio Digital de tipo I2S(SAI). Disponibilidad de RTC en carrier. Wiki variscite muy útil.
Aikri-X10-6S	-Nombre: Qualcomm® QCS610 -Fabricante: Qualcomm -Tipo: Qualcomm® Kryo™ 460 (basados en ARM) -Nº de cores: 8 (2 de alto rendimiento, 6 de bajo consumo) -Max CPU Clock: 2.2/1.8 -Real-Time Co-CPU: -	-RAM: 4GB LPDDR4 -eMMC: 16GB	-1 SD, 1 USB3.1, 2 USB2.0, 1 UART, 3 I2C, 2 SPI, GPIOs -CAN: - -Ethernet: 1 Gigabit Ethernet -WiFi: Qualcomm® Wi-Fi (WCN3980) 802.11a/b/g/n/ac - 2.4/5.0 GHz -Bluetooth: BT 5.0	Linux	Disponibilidad de comunicación vía GPS . Sensores: 1 acelerómetro y giróscopo, 1 magnetómetro, 1 proximidad, 1 temperatura y 1 presión. En cuanto a multimedia, tiene un controlador gráfico Adreno 608 , interfaz de cámara MIPI CSI, y respecto al audio un codificador Qualcomm WCD9341 y 3 entradas analógicas y digitales de micro. Sin embargo no disponen de comunicación CAN bus , a su vez tampoco dispone de un posible procesador para tareas en tiempo real .

Single Board Computers (SBC)					
Referencia de la SBC	CPU	Memoria y almacenamiento	Conectividad	Sistema Operativo	Comentarios
Digi ConnectCore 8X SBC Pro	-Nombre: NXP i.MX8QuadXPlus -Fabricante: NXP -Tipo: Cortex-A35 -Nº de cores: 4 -Max CPU Clock: 1.2 -Real-Time Co-CPU: Cortex-M4F a 266 MHz	-RAM: 2GB LPDDR4 -eMMC: 16GB	-1 MicroSD, 1 USB2.0, 1 USB3.0 en tipo-C, 2 UART, I2C, SPI, GPIOs, 1 JTAG -CAN: 2 -Ethernet: 2x Gigabit Ethernet 10/100/1000M -WiFi: 2 x 2 MIMO 802.11a/b/g/n/ac Seguridad: WEP, WPA-PSK/WPA2-Personal -Bluetooth: BT 5.0	Compatible con Yocto Project® Linux® y Android™	Dispone de 1 DSP HiFi4 de Tensilica, 1 botón de encendido y otro de reinicio, 2 leds de usuario y 1 de alimentación. Y salidas de potencia de 5V y 3.3V. Contiene dos conectores de antena U.FL en el módulo y un soporte de tarjetas con ranura Micro SIM. Seguridad mediante Digi TrustFence. Respecto a los gráficos, dispone de un motor de vídeo HD con capacidad para distintos flujos, reproducción de vídeo 3D en HD; por otro lado, comunicación 2xLVDS/MIPI para pantallas, 1 interfaz de cámara MIPI, y finalmente 2 entradas, 1 salida de audio, 1 entrada de micrófono y 1 conector jack de 3.5mm.
EIC-Q660-500	-Nombre: Qualcomm Snapdragon™ 660 -Fabricante: Qualcomm -Tipo: Qualcomm® Kryo™ 260 (basados en ARM) -Nº de cores: 8 -Max CPU Clock: 2.2 -Real-Time Co-CPU: -	-RAM: 6GB LPDDR4 -eMMC: 64GB	-1 MicroSD, 2 USB2.0, 1 USB3.1 en tipo-C, UART, 2 I2C, 2 SPI, 16 GPIOs, 1 JTAG -CAN: - -Ethernet: 1x 10/100 Mbps (Via USB2.0) -WiFi: IEEE 802.11 a/b/g/n/ac -Bluetooth: BT/BLE 5.1	Android 8.1	Sensores: 1 acelerómetro y giróscopo, 1 magnetómetro, 1 proximidad. Posibilidad de comunicación vía GPS con un Qualcomm GPS integrado . En cuanto a multimedia, hay una GPU Adreno 512 e interfaz de cámara con 2 ISP y 3 MIPI CSI de 4 líneas. 2 DSI para Display y para audio hay 2 MIC analógicos, 1 altavoz y 1 Jack de 3.5mm con MIC, entre otros. Sin embargo, no hay disponibilidad de una segunda CPU para tareas en tiempo real ni comunicación vía CAN bus .

Kits de Desarrollo					
Referencia del kit de desarrollo / SBC	Módulos compatibles y CPU	Multimedia	Interfaces y expansiones	Conectividad	Comentarios
Digi ConnectCore Kit de desarrollo 93	-Módulos compatibles: Digi ConnectCore® 93 -Nombre CPU: NXP® i.MX 93 -Fabricante CPU: NXP	-HDMI: - -Audio: 3x Synchronous Audio Interface (SAI) modules (up to 4 lanes) supporting I2S, AC97, TDM, codec/DSP and DSD interfaces -Interfaz cámara: 1x 1080p 60 MIPI-CSI (2-lane, 1.5 Gbps/lane)	-USB: 2x USB 2.0 OTG -High-Speed Expansions: SD: 1 interfaz uSDHC -Puertos Serie: -	-x8 UART, x8 I2C, x8 SPI, x4 PWM, 12 bit ADC (hasta 4 canales), hasta 112 GPIOs -CAN: 2 CAN/CAN-FD (Flexible Data-Rate) -Ethernet: 2x 10/100/1000M -WiFi: Wi-Fi 6 802.11ax dual-band 1x1 wireless -Bluetooth: BT 5.2	Entre las claves, beneficios y aplicaciones de este kit destacan el uso de plataforma SOM con i.MX 93 con dual-core, NPU para AI/ML Arm Ethos U65, WiFi de banda dual precertificada y Bluetooth 5.2, un alto nivel de compatibilidad de pines con otros Digi ConnectCore 8 SOMs. Construido sobre Digi TrustFence para seguridad y privacidad, monitorización remota con los servicios nube de Digi Connect Core y soporte de S.O. como Linux con Digi Embedded Yocto. (El kit incluye el SOM).
Symphony-Board	-Módulos compatibles: VAR-SOM-MX93, VAR-SOM-MX8, VAR-SOM-MX8X, VAR-SOM-MX8M-PLUS, VAR-SOM-MX8M-MINI, VAR-SOM-MX8M-NANO (no incluidos) -Nombre CPU: NXP® i.MX93, NXP® i.MX 8QuadMax / 8QuadPlus, NXP® i.MX 8X, NXP® i.MX 8M Plus, NXP® i.MX 8M Mini, NXP® i.MX8M Nano, NXP® i.MX8M Nano -Fabricante CPU: NXP	-HDMI: 2.0a - Necesita de la extensión VAR-EXT-HDMI -Audio: Headphone: conector 3.5mm, Line-in: conector 3.5mm, Digital Microphone: on-board -Interfaz cámara: Serial camera: Interfaz no incluida necesita la extensión (VCAM-5640S), Parallel camera: necesita de "camera extension board"	-USB: Host: 1 port, Type A connector, USB2.0; OTG: 1 port, Type C connector, USB3.0/2.0 TypeC -High-Speed Expansions: PCIe: conector mini-PCIe; SATA: SATA Connector; Tarjeta SD: SD Card Socket -Puertos Serie: Debug port: USB-SERIAL bridge, Micro USB type AB	-UART, I2C, SPI, JTAG (header en SoM), GPIOs, Timer, QSPI, ADC -CAN: Disponible -Ethernet: 2x 10/100/1000 Mbps, RJ45 -WiFi: On SoM -Bluetooth: On SoM	Symphony Board dispone de diversas ventajas entre las que destaca la capacidad de poder soportar la mayoría de SoM tratados de variscite que pertenecían a la familia Pin2Pin lo que permite una conexión sencilla; con gran cantidad de capacidad de comunicación. Para Display contiene a parte de HDMI, LVDS y DSI (para DP display port se necesitaría extensión), a su vez se dispone de touch panel tanto resistivo como capacitivo. Por otro lado, respecto a especificaciones electrónicas adicionales, se dispone de socket para Coin RTC Battery. Wiki variscite muy útil.
Aikri-X10-6D	-Módulos compatibles: Aikri-X10-6S -Nombre CPU: Qualcomm® QCS610 -Fabricante CPU: Qualcomm	-HDMI: - -Audio: Qualcomm® WCD9341 audio codec on-board, 1x Headset / Lineout, 3x Analog MIC input, 3x Digital MIC input -Interfaz cámara: 3x 4-Lane MIPI CSI	-USB: 1x USB 3.1, 2x USB 2.0 TypeC -High-Speed Expansions: SD: 1x SDC 3.0 -Puertos Serie: -	-x1 UART, x3 I2C, x2 SPI, x2 I2S, GPIOs(disponible en conector board-to-board) -CAN: - -Ethernet: 1x Gigabit Ethernet -WiFi: Qualcomm® Wi-Fi (WCN3980) 802.11a/b/g/n/ac - 2.4/5.0 GHz -Bluetooth: BT5.0	Disponibilidad de comunicación vía GPS. Respecto a multimedia habrá una GPU Adreno 608 , y por otro lado se dispondrá de una interfaz para Display DSI de 4 líneas MIPI-DSI entre otras. Sin embargo no hay disponibilidad de HDMI respecto a otras SBC que si que lo tienen, y a su vez no se puede realizar comunicación CAN bus . Kit contiene el SOM, la Carrier Board y una guía de comienzo rápido.

A.3. Anexo 3: Puesta en marcha del módulo VAR-SOM-MX93

El kit de desarrollo del módulo VAR-SOM-MX93 se compone un display LCD táctil y de la Symphony Board que se alimentará a 12V de corriente continua (DC). A su vez, se incluye sobre la SBC, el módulo VAR-SOM-MX93.

En primer lugar, y siguiendo la guía *Quick Start Guide* (disponible en la propia web de Variscite), se conectan el display LCD y el *Touch cable* a la SBC. Posteriormente, se procede a conectar el puerto micro USB con un PC externo para debug, y por último, se conecta la alimentación al *Jack* de entrada de 12V, con ello el hardware ya estaría en condiciones para ser encendido.

Antes de activar el switch que permite alimentar la placa, se ejecutará un *Host* desde el PC para poder hacer *debug* al procesador. En el caso de utilizar un PC con Sistema Operativo Windows se procede a instalar **Putty**, se trata de un SSH de código abierto con el que se podrá acceder al procesador.

Tras su instalación, se accede a él y se parametriza de la siguiente manera:

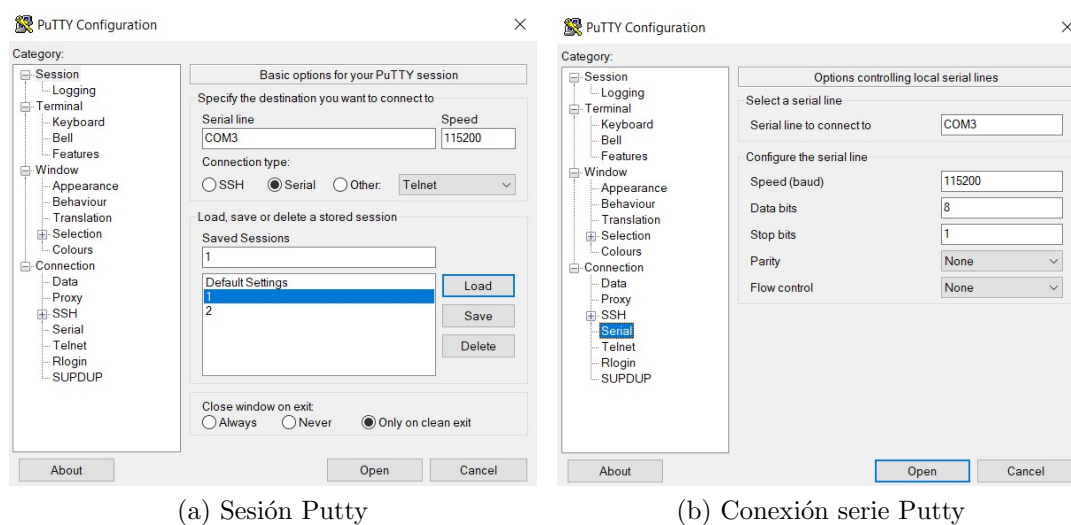


Figura A.22: Configuración del host Putty

Como se aprecia en la Figura A.22a y Figura A.22b, se accede mediante un puerto serie, con **Baud Rate: 115200**, **Data Bits: 8**, **Stop Bits: 1**, **Parity: None** y **Flow Control: None**. Acceder mediante SSH en vez de Serial es posible, sin embargo, para este caso se necesitará conocer la dirección IP del módulo.

Con el host Putty configurado, se abre la terminal y se espera a activar el switch. El entorno de Variscite dispone de dos métodos de arranque, desde la memoria flash eMMC, o desde una tarjeta microSD. El kit incluye una *microSD* pre-configurada con un Sistema Operativo Yocto, por lo que, al colocar el switch de modo de arranque en la posición *SD*, se arrancará desde ella. Por último, se activa el interruptor de arranque.

Por otro lado, si se utilizara como Sistema Operativo base uno distinto a Windows, por ejemplo en Ubuntu, podría utilizarse un terminal host picocom o minicom. Para

ello, utilizando una máquina virtual de Ubuntu, mediante el software *Virtual-Box* que permite crear y configurar máquinas virtuales, se procederá a abrir una terminal en la que se introducirán los siguientes comandos que permiten instalar e iniciar el host picocom:

```
$ sudo dmesg
```

Con este comando se pretende leer los últimos mensajes proporcionados por la CPU con el fin de encontrar el puerto USB utilizado para la conexión con la SBC (para poder usar ese puerto se tiene que añadir en la configuración de la máquina virtual). Como se observa en la Figura A.23, será el puerto USB *tttyUSB0*.

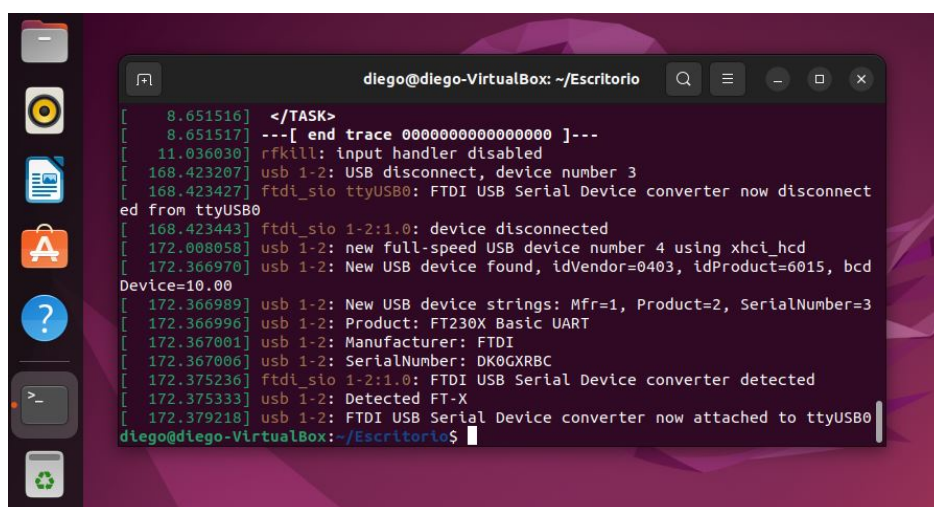
A screenshot of a terminal window titled 'diego@diego-VirtualBox: ~/Escritorio'. The terminal displays the output of the 'sudo dmesg' command, showing various system messages. Key messages include: 'usb 1-2: USB disconnect, device number 3', 'ftdi_sio tttyUSB0: FTDI USB Serial Device converter now disconnected', 'ftdi_sio 1-2:1.0: device disconnected', 'usb 1-2: new full-speed USB device number 4 using xhci_hcd', 'usb 1-2: New USB device found, idVendor=0403, idProduct=6015, bcdDevice=10.00', 'usb 1-2: New USB device strings: Mfr=1, Product=2, SerialNumber=3', 'usb 1-2: Product: FT230X Basic UART', 'usb 1-2: Manufacturer: FTDI', 'usb 1-2: SerialNumber: DK0CXRBC', 'ftdi_sio 1-2:1.0: FTDI USB Serial Device converter detected', 'usb 1-2: Detected FT-X', and 'usb 1-2: FTDI USB Serial Device converter now attached to tttyUSB0'. The prompt 'diego@diego-VirtualBox:~/Escritorio\$' is visible at the bottom.

Figura A.23: Mensajes recibidos mediante comando *sudo dmesg* en Ubuntu

```
$ sudo apt-get install picocom
```

```
$ sudo picocom -b 115200 /dev/ttyUSBX
```

Se instala picocom utilizando *sudo* que dará permisos y acceso de administrador. Finalmente se lanza picocom mediante este último comando, sustituyendo la X por el número apropiado.

Debido a que el Sistema Operativo base del PC a utilizar es Windows, se decide por el uso del host Putty, siguiendo los pasos que se han nombrado anteriormente.

A.3.1. Yocto OS: tests y uso de la interfaz

Al arrancar inicialmente el módulo desde el host Putty en modo SD, se muestran en la terminal los mensajes de comienzo de Yocto y tras ello se solicita el nombre de usuario, que será *root* para el uso de permisos y acceso como administrador.

Para instalar el Yocto en el almacenamiento interno, tan solo será necesario escribir la siguiente línea en la terminal y seguir las instrucciones que se pidan por pantalla:

```
# install_yocto.sh
```

Después de esto, se podrá reiniciar el módulo tras cambiar el modo de arranque en la SBC a *INT* (modo interno). Ya desde el almacenamiento interno, se puede observar

el arranque correcto de Yocto.

Variscite ofrece la posibilidad de crear una imagen propia de Yocto siguiendo los pasos que se pueden encontrar en la wiki de Variscite. Sin embargo, esta opción requiere del uso de la máquina virtual Ubuntu, en la que se instala la imagen. Este proceso necesita de una gran cantidad de recursos que esta máquina virtual no es capaz de suministrar. Tras intentar crear una imagen Yocto sin resultados positivos, se escoge la opción más óptima y rápida, siendo el uso de la imagen ya prediseñada de la microSD.

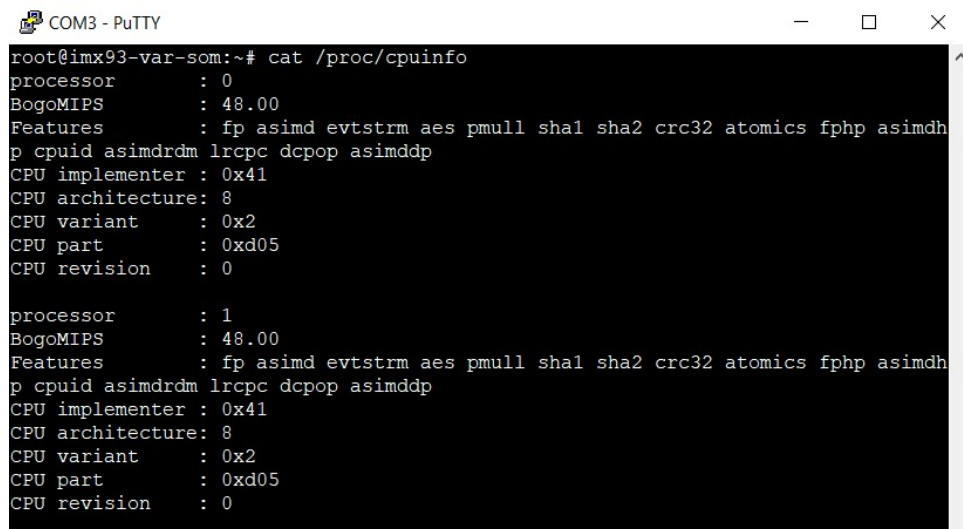
Para comenzar con el hardware, se realizan distintos tests de funcionamiento y uso de la interfaz que se pueden encontrar en la wiki de Variscite[35]. Con ellos se podrán probar distintas características del módulo y de la interfaz.

- *CPU number of cores:*

```
# echo 0 >/sys/devices/system/cpu/cpuX/online
# echo 1 >/sys/devices/system/cpu/cpuX/online
```

Mediante estos comandos se consigue manipular el número de cores activos. Al realizar *echo 0* sobre ese directorio se deshabilita el core X, mientras que con *echo 1* se habilita. Este SOM dispone de dos cores 0 y 1, el primero siempre se encuentra habilitado, pudiendo por tanto actuar sobre el segundo. Para ver los núcleos activos se puede ver utilizar el siguiente comando:

```
# cat /proc/cpuinfo
```



```
COM3 - PuTTY
root@imx93-var-som:~# cat /proc/cpuinfo
processor       : 0
BogoMIPS      : 48.00
Features      : fp asimd evtstrm aes pmull sha1 sha2 crc32 atomics fphp asimdhp
p cpuid asimdrdm lrcpc dcpop asimddp
CPU implementer : 0x41
CPU architecture: 8
CPU variant    : 0x2
CPU part       : 0xd05
CPU revision   : 0

processor       : 1
BogoMIPS      : 48.00
Features      : fp asimd evtstrm aes pmull sha1 sha2 crc32 atomics fphp asimdhp
p cpuid asimdrdm lrcpc dcpop asimddp
CPU implementer : 0x41
CPU architecture: 8
CPU variant    : 0x2
CPU part       : 0xd05
CPU revision   : 0
```

Figura A.24: Cores activos de la CPU

Como se aprecia en la anterior figura, se encuentran ambos cores del i.MX93 activos.

- *CPU temperature:*

El SOC i.mx93 dispone de un sensor de temperatura interno que puede ser leído y que se muestra en **miligrados Celsius**. El acceso a ese valor será mediante:

```
# cat /sys/class/thermal/thermal_zone0/temp
```

```

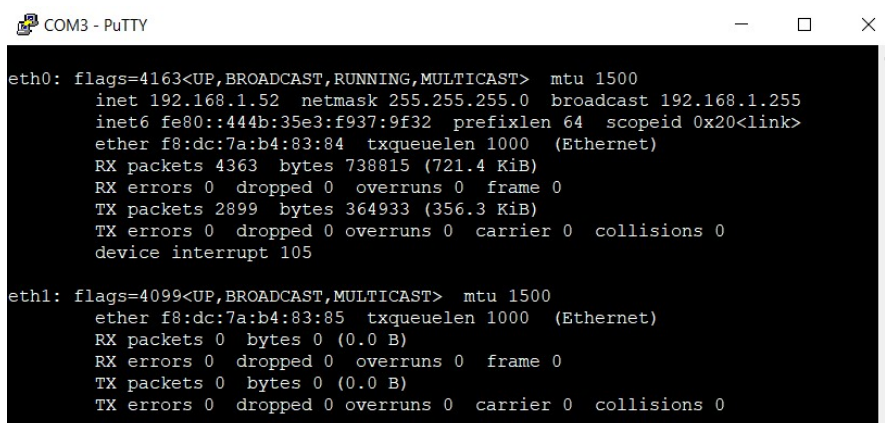
root@imx93-var-som:~# cat /sys/class/thermal/thermal_zone0/temp
55850
root@imx93-var-som:~# █

```

Figura A.25: Temperatura de la CPU medida en miligrados Celsius

- *Tests de actuación de Ethernet:*

Para la prueba de la conexión Ethernet ,en primer lugar, se escribe el comando # `ifconfig` que permitirá conocer la dirección IP asignada al módulo. En la siguiente figura se muestran los dos puertos Ethernet, y en *eth0* (el puerto conectado), la dirección IP asignada es **192.168.1.52**.



```

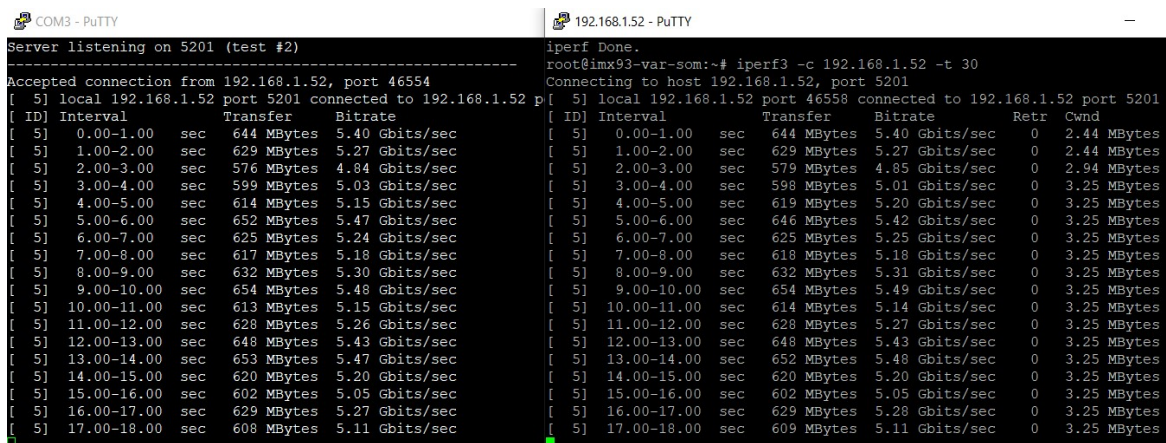
COM3 - PuTTY
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.52 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::444b:35e3:f937:9f32 prefixlen 64 scopeid 0x20<link>
    ether f8:dc:7a:b4:83:84 txqueuelen 1000 (Ethernet)
    RX packets 4363 bytes 738815 (721.4 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2899 bytes 364933 (356.3 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 105

eth1: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether f8:dc:7a:b4:83:85 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Figura A.26: Información de la comunicación Ethernet del módulo

A continuación, # `iperf3 -s` permitirá escuchar lo que envíe un cliente. Abriendo otra terminal Putty como cliente, se introduce en ella para realizar un test TCP durante 30 segundos, el comando \$ `iperf3 -c DIR_IP_SERVER -t 30`



```

COM3 - PuTTY
Server listening on 5201 (test #2)
Accepted connection from 192.168.1.52, port 46554
[ ID] Interval      Transfer      Bitrate
[ 5] 0.00-1.00 sec  644 MBytes  5.40 Gbits/sec
[ 5] 1.00-2.00 sec  629 MBytes  5.27 Gbits/sec
[ 5] 2.00-3.00 sec  576 MBytes  4.84 Gbits/sec
[ 5] 3.00-4.00 sec  599 MBytes  5.03 Gbits/sec
[ 5] 4.00-5.00 sec  614 MBytes  5.15 Gbits/sec
[ 5] 5.00-6.00 sec  652 MBytes  5.47 Gbits/sec
[ 5] 6.00-7.00 sec  625 MBytes  5.24 Gbits/sec
[ 5] 7.00-8.00 sec  617 MBytes  5.18 Gbits/sec
[ 5] 8.00-9.00 sec  632 MBytes  5.30 Gbits/sec
[ 5] 9.00-10.00 sec 654 MBytes  5.48 Gbits/sec
[ 5] 10.00-11.00 sec 613 MBytes  5.15 Gbits/sec
[ 5] 11.00-12.00 sec 628 MBytes  5.26 Gbits/sec
[ 5] 12.00-13.00 sec 648 MBytes  5.43 Gbits/sec
[ 5] 13.00-14.00 sec 653 MBytes  5.47 Gbits/sec
[ 5] 14.00-15.00 sec 620 MBytes  5.20 Gbits/sec
[ 5] 15.00-16.00 sec 602 MBytes  5.05 Gbits/sec
[ 5] 16.00-17.00 sec 629 MBytes  5.27 Gbits/sec
[ 5] 17.00-18.00 sec 608 MBytes  5.11 Gbits/sec

192.168.1.52 - PuTTY
iperf Done.
root@imx93-var-som:~# iperf3 -c 192.168.1.52 -t 30
Connecting to host 192.168.1.52, port 5201
[ ID] Interval      Transfer      Bitrate      Retr  Cwnd
[ 5] 0.00-1.00 sec  644 MBytes  5.40 Gbits/sec  0  2.44 MBytes
[ 5] 1.00-2.00 sec  629 MBytes  5.27 Gbits/sec  0  2.44 MBytes
[ 5] 2.00-3.00 sec  579 MBytes  4.85 Gbits/sec  0  2.94 MBytes
[ 5] 3.00-4.00 sec  598 MBytes  5.01 Gbits/sec  0  3.25 MBytes
[ 5] 4.00-5.00 sec  619 MBytes  5.20 Gbits/sec  0  3.25 MBytes
[ 5] 5.00-6.00 sec  646 MBytes  5.42 Gbits/sec  0  3.25 MBytes
[ 5] 6.00-7.00 sec  625 MBytes  5.25 Gbits/sec  0  3.25 MBytes
[ 5] 7.00-8.00 sec  618 MBytes  5.18 Gbits/sec  0  3.25 MBytes
[ 5] 8.00-9.00 sec  632 MBytes  5.31 Gbits/sec  0  3.25 MBytes
[ 5] 9.00-10.00 sec 654 MBytes  5.49 Gbits/sec  0  3.25 MBytes
[ 5] 10.00-11.00 sec 614 MBytes  5.14 Gbits/sec  0  3.25 MBytes
[ 5] 11.00-12.00 sec 628 MBytes  5.27 Gbits/sec  0  3.25 MBytes
[ 5] 12.00-13.00 sec 648 MBytes  5.43 Gbits/sec  0  3.25 MBytes
[ 5] 13.00-14.00 sec 652 MBytes  5.48 Gbits/sec  0  3.25 MBytes
[ 5] 14.00-15.00 sec 620 MBytes  5.20 Gbits/sec  0  3.25 MBytes
[ 5] 15.00-16.00 sec 602 MBytes  5.05 Gbits/sec  0  3.25 MBytes
[ 5] 16.00-17.00 sec 629 MBytes  5.28 Gbits/sec  0  3.25 MBytes
[ 5] 17.00-18.00 sec 609 MBytes  5.11 Gbits/sec  0  3.25 MBytes

```

Figura A.27: Comunicación Ethernet test TCP

- *Wireless / WiFi:*

En primer lugar se comprueba su estado y disponibilidad. Para habilitarla puede usarse el segundo comando:

```

# nmcli dev show wlan0
# nmcli radio wifi on.

```

A continuación se pasa a configurar el cliente, leyendo la lista de posibles APs y posteriormente conectándose a la red:

```
# nmcli dev wifi list.
```

```
# nmcli dev wifi connect <SSID> password <PASSWORD> ifname wlan0
```

Por último, si se quiere conocer el estado de la conexión pueden ejecutar los siguientes comandos. En la figura se utiliza el segundo, mostrando las conexiones activas:

```
# nmcli con show SSID;
```

```
# nmcli con show
```

```
root@imx93-var-som:~# nmcli con show
NAME                                UUID                                TYPE    DEVICE
Supervisor eth0                    9a0191f1-2c78-362a-9170-8792cac495a6 ethernet eth0
lo                                  2c2a9518-24df-4598-9e09-4126c491df68 loopback lo
MOVISTAR_PLUS_5B76                 e725eb34-c396-4f39-884c-1916d64e88f4 wifi     wlan0
Wired connection 1                  59002dc8-0f07-335d-9047-9e993d2f5774 ethernet --
root@imx93-var-som:~#
```

Figura A.28: Comunicación con WiFi

Se dispone de mayor capacidad de configuración y personalización, no obstante, no se ha profundizado más en ello.

A.3.2. Instalación de Debian

Para la instalación de Debian se sigue el mismo procedimiento que se había realizado previamente con Yocto. Se descargará una imagen de Debian en una tarjeta SD y con ella se grabará el Sistema Operativo en el almacenamiento interno del SOM.

Para descargar esta imagen de Debian, se accede al servidor de Variscite FTP[36], entrando con las credenciales aportadas por Variscite[35].

Dentro del server se busca la carpeta del SOM a tratar y se descarga el archivo con nombre: *mx93-debian-bookworm-6.1.1_1.0.1-v1.0.img.zst*.

Finalmente se necesita un software que permita instalar la imagen en una tarjeta SD. Se utiliza Win32 Disk Imager que permite tal objetivo, y con el que se graba la imagen descomprimida sobre una tarjeta SD.

Tras la instalación, se puede insertar la tarjeta en el lector de la SBC y arrancar desde la SD. Una vez el arranque se complete, se introduce el nombre de usuario (**root** para obtener permisos de administrador) y contraseña (también **root**). Finalmente, se instalará Debian en el almacenamiento interno del SOM:

```
# install_debian.sh
```

Por último, se puede volver a arrancar cambiando el modo a *INT*, ahora mediante la eMMC, y realizar los mismos tests u otros, como ocurría en Yocto o instalar el software de automatización Home Assistant.

A.4. Anexo 4: Instalación y puesta en marcha de Home Assistant

En este anexo se va a realizar el proceso de instalación e inicio de Home Assistant, argumentando las tomas de decisiones y su procedimiento.

Teniendo un hardware utilizado basado en un Sistema Operativo Linuxy una interfaz que ofrece es básicamente textual, complica en determinada manera la instalación rápida de métodos como *Home Assistant Operating System* o *Home Assistant Container*, teniendo que acceder a comandos distintos para llevarlos a cabo.

Es por ello que en una primera instancia, se procede a instalar ***Home Assistant Core*** siguiendo los pasos indicados en la web de Home Assistant [45].

Sobre el host Putty se instalarán los paquetes necesarios y el entorno virtual de Home Assistant Core. Antes de comenzar, el sistema debe estar actualizado.

```
# sudo apt-get update
# sudo apt-get upgrade -y
```

Con el sistema ya actualizado, se podrá instalar las dependencias necesarias para el uso de Home Assistant:

```
# sudo apt-get install -y python3 python3-dev python3-venv
python3-pip bluez libffi-dev libssl-dev libjpeg-dev zlib1g-dev autoconf
build-essential libopenjp2-7 libtiff5 libturbojpeg0-dev tzdata ffmpeg
liblapack3 liblapack-dev libatlas-base-dev
```

Home Assistant sugiere crear un usuario de homeassistant. Sin embargo, si se quiere acceder a distintos periféricos de la SBC o características necesarias, es importante que el usuario activo sea el administrador del sistema ***root***.

Por tanto, no se deberá crear y activar el usuario *homeassistant*, sino que se crea el directorio de instalación de homeassistant y se pasa a este mediante el usuario root. Tras ello se crea y activa el entorno virtual.

```
# mkdir /srv/homeassistant
# cd /srv/homeassistant
# python3 -m venv .
# source bin/activate
```

Una vez el entorno virtual ha sido activado la línea de comandose se verá como: (homeassistant) root@imx93-var-som:/srv/homeassistant #. Solo quedará instalar el paquete Python requerido, y con ello Home Assistant Core.

```
# python3 -m pip install wheel
# pip3 install homeassistant==2023.8.3
```

Finalmente se da comienzo a Home Assistant Core, completando de esta forma la

instalación y creando un directorio de configuración:

```
# hass.
```

Con ello ya se podrá acceder a la interfaz web de Home Assistant en la dirección ***http://<IP_ADDRESS>:8123***, siendo esta la IP del SOM.

Home Assistant Core es una buena versión, sin embargo tiene una serie de defectos. En primer lugar no permite el uso de complementos o Add-ons, que son muy útiles y hacen que Home Assistant sea más completo y sencillo de utilizar. No obstante, como mayor defecto, se encuentra que para realizar configuraciones o crear automatizaciones será necesario modificar los archivos `configuration.yaml` y otros que se encuentren en el directorio de configuración, siendo fundamental acceder mediante el host Putty al contenido de estos, y haciendo que el procedimiento sea algo tedioso.

Por ello, se decide instalar la versión similar pero más completa ***Home Assistant Supervised***. Aunque su proceso de instalación sea algo más complejo, su resultado es más completo y agradable para un uso posterior.

En primer lugar, este método tiene requerimientos muy estrictos, tan solo soportando Debian en el caso de Linux. En este caso, al estar usando un S.O. Debian esta instalación es posible. Con esto claro, se accede al proceso de instalación [48]. El supervisor es un dispositivo completo que administrará todo el sistema. El usuario será responsable de que los componentes necesarios estén instalados y mantenidos.

Se comienza instalando las siguientes dependencias necesarias:

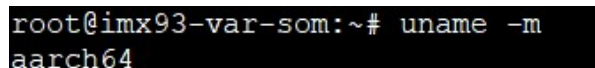
```
# sudo apt install apparmor jq wget curl udisks2 libglib2.0-bin  
network-manager dbus lsb-release systemd-journal-remote systemd-resolved  
-y
```

A continuación se instala entorno de desarrollo de contenedores Docker-CE:

```
# sudo curl -fsSL get.docker.com | sh
```

El siguiente paso es instalar el *OS-Agent*, que es la forma en la que el supervisor maneja tareas adicionales en el host [45]. Para ello, se descarga el adecuado a la arquitectura del procesador que se puede conocer mediante: [49]

```
# uname -m
```



```
root@imx93-var-som:~# uname -m  
aarch64
```

Figura A.29: Arquitectura del kernel Linux instalado

Con ello se instala el paquete descargado de la siguiente manera y se comprueba que la instalación haya sido satisfactoria:

```
# sudo dpkg -i os-agent_1.0.0_linux_aarch64.deb  
  
# gdbus introspect --system --dest io.hass.os --object-path /io  
/hass/os.
```

El último paso será instalar el paquete Debian de Home Assistant Supervised:

```
# wget https://github.com/home-assistant/supervised-installer
/releases/latest/download/homeassistant-supervised.deb apt install
./homeassistant-supervised.deb
```

A.5. Anexo 5: Instalación de broker mosquitto y configuración en Dingtian

El módulo de relés permite una amplia variedad de opciones para poder controlar mediante su ESP32 las distintas entradas y relés de los que dispone permitiendo comunicarse con este Home Assistant mediante **MQTT**. [43]

Por tanto, hará falta comunicar vía MQTT el SOM con el módulo de relés, siendo necesario un broker que facilite esta conexión. Se instala de este modo un broker **mosquitto** en el SOM. Existen dos opciones, a partir de un complemento en Home Assistant o mediante la terminal Putty. Por seguridad de comunicación entre el SOM y el módulo se decide la segunda opción. En primer lugar, se comprueba que el sistema esté actualizado completamente y a continuación se instala mosquitto:

```
# sudo apt install mosquitto

# sudo systemctl enable mosquitto

# sudo systemctl status mosquitto
```

Mediante el primer comando se instala el broker mosquitto, a continuación se habilita y finalmente se comprueba su estado.[51]

```
root@imx93-var-som:~# sudo systemctl status mosquitto
● mosquitto.service - Mosquitto MQTT Broker
   Loaded: loaded (/lib/systemd/system/mosquitto.service; enabled; preset: en>
   Active: active (running) since Mon 2023-08-21 18:49:03 UTC; 8h ago
     Docs: man:mosquitto.conf(5)
           man:mosquitto(8)
   Process: 727 ExecStartPre=/bin/mkdir -m 740 -p /var/log/mosquitto (code=exi>
   Process: 731 ExecStartPre=/bin/chown mosquitto /var/log/mosquitto (code=exi>
   Process: 733 ExecStartPre=/bin/mkdir -m 740 -p /run/mosquitto (code=exited,>
   Process: 739 ExecStartPre=/bin/chown mosquitto /run/mosquitto (code=exited,>
  Main PID: 742 (mosquitto)
    Tasks: 1 (limit: 2140)
   Memory: 2.2M
      CPU: 10.281s
  CGroup: /system.slice/mosquitto.service
          └─742 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf
```

Figura A.30: Estado del broker mosquitto tras la instalación

Con el broker instalado, se configuran en la interfaz del módulo de relés los parámetros para la comunicación MQTT, siendo la dirección del broker la del SOM.

MQTT	Head slash("/") ☒	Broker Address	Broker Port	Broker Username	Broker Password
	QoS 1 (At least once) v	192.168.1.52	1883	mqtt	...
	Retain ☒				
	KeepAlive 120 S				
	MFR dingtian				
	Area				

Figura A.31: Configuración del broker mosquitto en la interfaz del módulo de relés

A.6. Anexo 6: Instalación de MCUXpresso y aplicación detallada

Para comenzar a utilizar el Cortex-M33, será necesario poder tener uso de un entorno con el que poder programarlo, construir posteriormente el proyecto realizado y finalmente ejecutarlo. Es por ello que Variscite ofrece el uso de una versión de **FreeRTOS**, que se trata de un sistema operativo para tiempo real. La versión será un SDK de **MCUXpresso**, que se encuentra diseñado para su uso en dispositivos NXP.

A partir de la wiki de Variscite [35] se consigue el procedimiento de instalación y uso de distintas demostraciones que sirven para probar el co-procesador con MCUXpresso.

En primer lugar será necesario usar un entorno base Linux, se utilizará una máquina Virtual Ubuntu. Estando esta ya configurada, se podrá proceder a instalar directamente los paquetes requeridos.

Para comenzar, en una terminal se instala el paquete cmake:

```
$ sudo apt-get install cmake
```

Tras ello se crea un directorio *var-mcuexpresso* en el que se descarga y posteriormente instala la cadena de herramientas básicas GNU-ARM:

```
$ wget https://developer.arm.com/-/media/Files/downloads/gnu-rm/10.3-2021.10/gcc-arm-none-eabi-10.3-2021.10-x86_64-linux.tar.bz2
```

```
$ tar xvf gcc-arm-none-eabi-10.3-2021.10-x86_64-linux.tar.bz2
```

Para terminar el proceso de instalación, se descarga el MCUXpresso SDK para el SOM y se accede a la carpeta de freertos-variscite generado:

```
$ git clone https://github.com/varigit/freertos-variscite -b mcuxpresso_sdk_2.13.x-var02
```

```
$ cd freertos-variscite
```

Una vez instalado, se pueden encontrar distintas demos a probar. De entre ellas se prueba una que permite parpadear un led ya que permite el uso de GPIOs.

Para comenzar a usar esta demo, esta se encuentra configurada totalmente y no hay que programar nada. Por lo tanto, se comienza construyendo el proyecto que luego se ejecutará en el co-procesador. Variscite ofrece distintas formas de construir un proyecto, eligiendo la opción manual y accediendo al directorio de la demo. Una vez ahí se accede a la carpeta de construcción:

```
$ cd armgcc
```

A continuación se exportan al directorio las herramientas de GNU-ARM instaladas previamente y se construye finalmente:

```
$ export ARMGCC_DIR= /var-mcuexpresso/gcc-arm-none-eabi-10.3-2021.10
```

```
$ ./build_all.sh >/dev/null
```

Tras esto se genera un archivo `.elf` en una carpeta de debug, que será la imagen del programa que se enviará al SOM y que a continuación se ejecuta en él. También se pueden encontrar distintas opciones para ejecutar un programa en el SOM, sin embargo, se decide desde Linux. Se comienza incrementando el nivel de registro del kernel mientras debug y se apaga el co-procesador en el caso de que este activo.

```
# sysctl kernel.printk=7;
```

```
# echo stop >/sys/class/remoteproc/remoteproc0/state
```

Con el procesador parado, se carga el *firmware* en él y finalmente se activa:

```
# echo stop >/sys/class/remoteproc/remoteproc0/state
```

```
# echo start >/sys/class/remoteproc/remoteproc0/state
```

Para probar que el comportamiento es adecuado se conecta un LED en el pin indicado en la demo y se observa un parpadeo que se enciende y apaga cada segundo.

Apéndice B

Bibliografía

- [1] Mahmoud H. Elkholy, Tomonobu Senjyu, Mohammed Elsayed Lotfy, Abdelrahman Elgarhy, Nehad S. Ali, Tamer S. Gaafar, *Design and Implementation of a Real-Time Smart Home Management System Considering Energy Saving*, <https://www.mdpi.com/2071-1050/14/21/13840>, 2022, accessed: 2023
- [2] Bilal Mubdir, Asaad Al-Hindawi, Noor Hadi, *Design of Smart Home Energy Management System for Saving Energy*, <https://eujournal.org/index.php/esj/article/view/8350>, 2016, accessed: 2023
- [3] Irshad Hussain, Majid Ullah, Ibrar Ullah, Asima Bibi, Muhammad Naeem, Madhusudan Singh, Dhananjay Singh, *Optimizing Energy Consumption in the Home Energy Management System via a Bio-Inspired Dragonfly Algorithm and the Genetic Algorithm*, <https://www.mdpi.com/2079-9292/9/3/406>, 2020, accessed: 2023
- [4] U.S. Energy Information Administration, *Energy use in homes*, <https://www.eia.gov/energyexplained/use-of-energy/homes.php>, 2021, accessed: 2023
- [5] Grön Circular Company, *USO DE ENERGÍAS RENOVABLES EN LA CONSTRUCCIÓN*, <https://www.groncr.com/uso-de-energias-renovables-en-la-construccion/>, accessed: 2023
- [6] Enel Green Power, *Todas las ventajas de la energía solar*, <https://www.enelgreenpower.com/es/learning-hub/energias-renovables/energia-solar/ventajas-energia-solar>, accessed: 2023
- [7] Repsol, *¿Cuántos kWh consume una casa al mes?*, <https://www.repsol.es/particulares/asesoramiento-consumo/cuantos-kwh-consume-casa/>, accessed: 2023
- [8] Danahé San Juan, *Instalación eléctrica de equipos HVACR*, <https://0grados.com/instalacion-electrica-de-equipos-hvacr/>, 2012, accessed: 2023
- [9] Revista Electro Industria, *Protocolos de Comunicaciones para Automatización de Edificios*, <https://www.emb.cl/electroindustria/articulo.mvc?xid=1223&ni=protocolos-de-comunicaciones-para-automatizacion-de-edificios>, 2023, accessed: 2023

- [10] Iberdrola, *Los contadores inteligentes, una pieza básica para la digitalización de la red*, <https://www.iberdrola.com/innovacion/contadores-inteligentes>, accessed: 2023
- [11] Ivy Metering, *Protocolo de comunicación del medidor eléctrico.*, <https://www.ivy-emeter.com/html/companynews-show-251.html>, 2020, accessed: 2023
- [12] Junkers, *JUNKERS Calentador electrico instantaneo ED 21 3S*, <https://www.solotermos.es/junkers-termos-electricos-instantaneos-junkers\-calentador-electrico-instantaneo-ed-21-3s-p-667.html>, accessed: 2023
- [13] Soloelectronicos, *Convierta su calentador eléctrico en inteligente*, <https://soloelectronicos.com/2018/12/09/convierta-su-calentador-electrico-en-inteligente/>, 2018, accessed: 2023
- [14] Grupo Lara, *CALENTADOR ELÉCTRICO COMPACT ELECTRICHEAT ATRALPOOL*, <https://piscinas-lara.com/tienda/calentador-electrico-compact-electricheat-atralpool/>, accessed: 2023
- [15] Suministros del Sol, *Paneles Solares*, <https://suministrosdelsol.com/es/12-paneles-solares>, accessed: 2023
- [16] Marcos Carbonell, Hogarsense, *El calentador solar fotovoltaico*, <https://www.hogarsense.es/placas-solares/calentador-solar>, 2023, accessed: 2023
- [17] Javier Navarro, *Monitoreando la temperatura de un termotanque solar usando D1 Mini y Wifi*, <https://acortar.link/ut5e15>, 2018, accessed: 2023
- [18] Solatom, *Módulos Fresnel individuales para investigación y prototipos*, <https://solatom.com/investigacion/>, accessed: 2023
- [19] Monsolar, *INVERSORES SOLARES*, <https://www.monsolar.com/fotovoltaica-aislada/inversores.html>, accessed: 2023
- [20] Félix Sasián, Ricardo Theron, Diego Gachet, *Protocolo para comunicación inalámbrica en instalaciones de energías renovables*, https://www.researchgate.net/publication/305112081_Protocolo_para_comunicacion_inalambrica_en_instalaciones_de_energias_renovables, 2016, accessed: 2023
- [21] SMA, *Optimización del rendimiento directamente desde el inversor*, <https://www.sma.de/es/productos/caracteristicas-producto-interfaces/interfaz-de-protocolo-modbus>, accessed: 2023
- [22] Autosolar, *Cómo cargar una batería con un panel solar*, <https://autosolar.pe/aspectos-tecnicos/como-cargar-una-bateria-con-un-panel-solar>, accessed: 2023
- [23] BBVA, *¿Es posible cargar un coche eléctrico con paneles solares?*, <https://www.bbva.com/es/sostenibilidad/es-posible-cargar-un-coche-electrico-con-paneles-solares/>, accessed: 2023

- [24] tritekbattery, *CAN Bus: Sistema de Gestión de Baterías y Comunicaciones*, <https://tritekbattery.com/es/can-bus-battery-management-and-communications-system/>, accessed: 2023
- [25] Red Hat, *¿Qué es el Internet de las cosas (IoT)?*, <https://www.redhat.com/es/topics/internet-of-things/what-is-iot>, 2023, accessed: 2023
- [26] Wago, *Rápida comunicación entre dispositivos de campo y automatización: MOD-BUS*, <https://www.wago.com/es/modbus>, 2020, accessed: 2023
- [27] Ingeniería y Mecánica Automotriz, *¿Qué es el CAN BUS y cómo funciona?*, <https://www.ingenieriaymecanicaautomotriz.com/que-es-el-can-bus-y-como-funciona/>, 2020, accessed: 2023
- [28] Beatriz Velasco, Laura Burón, Rodrigo San Martín, *Wifi, qué es y como funciona nuestra red inalámbrica wifi*, <https://roams.es/companias-telefonicas/blog/tecnologia/wifi-que-es/>, 2023, accessed: 2023
- [29] Rocio GR, *Qué es el Bluetooth y todo lo que permite esta tecnología*, <https://www.adslzone.net/reportajes/tecnologia/bluetooth/>, 2022, accessed: 2023
- [30] Venco, *Qué es ZigBee, cómo funciona y características principales*, <https://www.vencoel.com/que-es-zigbee-como-funciona-y-caracteristicas-principales/>, 2020, accessed: 2023
- [31] Antiun, *Protocolo IP*, <https://www.antiun.com/protocolo-ip/>, accessed: 2023
- [32] Microsoft Azure, *¿Qué es la nube?*, <https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-the-cloud>, accessed: 2023
- [33] Variscite, *VAR-SOM-MX93 : NXP i.MX93*, <https://www.variscite.com/product/system-on-module-som/cortex-a55/var-som-mx93-nxp-i-mx-93/>, accessed: 2023
- [34] Variscite, *Symphony-Board*, <https://www.variscite.com/product/single-board-computers/symphony-board/>, accessed: 2023
- [35] Variscite, *Wiki-VAR-SOM-MX93*, <https://variwiki.com/index.php?title=VAR-SOM-MX93>, accessed: 2023
- [36] Variscite, *Variscite's FTP server*, <https://ftp.variscite.com>, accessed: 2023
- [37] SONOFF, *ZIGBEE 3.0 USB DONGLE PLUS*, <https://sonoff.tech/product/gateway-and-sensors/sonoff-zigbee-3-0-usb-dongle-plus-e/>, accessed: 2023
- [38] SONOFF, *SONOFF S26R2ZB ZigBee Smart Plug*, <https://sonoff.tech/product/s26r2zb/>, accessed: 2023
- [39] SONOFF, *SONOFF SNZB-01 ZigBee Wireless Switch*, <https://sonoff.tech/product/gateway-and-sensors/snz-01/>, accessed: 2023

- [40] SONOFF, *SONOFF SNZB-04 ZigBee Wireless Door/Window Sensor*, <https://sonoff.tech/product/gateway-and-sensors/snzb-04/>, accessed: 2023
- [41] Remihome, *Aqara – Sensor de movimiento – RTCGQ01LM*, <https://remihome.es/dispositivos/aqara-sensor-de-movimiento-rtcgq01lm/>, accessed: 2023
- [42] SONOFF, *SONOFF Zigbee LCD Smart Temperature Humidity Sensor*, <https://sonoff.tech/product/gateway-and-sensors/snzb-2d/>, accessed: 2023
- [43] Dingtian, *Dingtian-tech*, https://www.dingtian-tech.com/en_us/relay16.html, accessed: 2023
- [44] ARM, *Cortex-M33*, <https://developer.arm.com/Processors/Cortex-M33>, accessed: 2023
- [45] Home Assistant, *Home Assistant*, <https://www.home-assistant.io/>, accessed: 2023
- [46] Charly Pascual, Programarfacil, *Home Assistant instalación y primeros pasos*, <https://programarfacil.com/domotica/home-assistant/>, accessed: 2023
- [47] Steve Emms, LinuxLinks, *11 Best Free and Open Source Solutions for Home Automation*, <https://www.linuxlinks.com/automation/>, 2022, accessed: 2023
- [48] ikifar2012 *Home-Assistant Supervised-installer*, <https://github.com/home-assistant/supervised-installer>, 2019, accessed: 2023
- [49] apuntes.de *Comando uname de Linux*, <https://apuntes.de/linux-certificacion-lpi/el-comando-uname/#gsc.tab=0>, accessed: 2023
- [50] Mia.Ma, SONOFF *How to Use SONOFF Dongle Plus on Home Assistant?*, <https://sonoff.tech/product-review/how-to-use-sonoff-dongle-plus-on-home-assistant-how-to-flash-firmware/>, 2021, accessed: 2023
- [51] HowtoForge *Cómo instalar el gestor de mensajes MQTT de Mosquitto en Debian 11*, <https://howtoforge.es/como-instalar-el-gestor-de-mensajes-mqtt-de-mosquitto-en-debian-11/>, 2023, accessed: 2023
- [52] domoticasolar *ModBus TCP. Obtén los datos de tu Inversor Solar. Node Red. Ingeteam*, <https://domotica.solar/modbus-tcp-obten-los-datos-de-tu-inversor-solar-node-red-ingeteam>, 2023, accessed: 2023
- [53] iammeter *Support Modbus/TCP in the Wi-Fi energy meter of IAMMETER*, <https://www.iammeter.com/newsshow/news-modbus-tcp-energy-meter>, accessed: 2023
- [54] Intesis *Mitsubishi Electric Centralized Controller TCP/ IP*, https://www.intesis.com/docs/librariesprovider11/manuals-design-guides/user-manuals/ac-interfaces/user-manual-inmbsmittxxxxc000.pdf?sfvrsn=e23150d7_20, accessed: 2023

- [55] Solarfocus *boiler control: Modbus TCP - Register data*, <https://acortar.link/PWWAzo>, accessed: 2023
- [56] OpenHAB *OpenHAB*, <https://www.openhab.org/>, accessed: 2023
- [57] Domoticz *Domoticz*, https://www.domoticz.com/wiki/Main_Page, accessed: 2023
- [58] RexControls *Rexcontrols*, <https://www.rexcontrols.com/>, accessed: 2023
- [59] Codesys *Codesys*, <https://www.codesys.com/>, accessed: 2023
- [60] OpenRemote *OpenRemote*, <https://www.openremote.io/>, accessed: 2023
- [61] EdgeXFoundry *EdgeXFoundry*, <https://www.edgexfoundry.org/>, accessed: 2023
- [62] ROS *ROS*, <https://www.ros.org/>, accessed: 2023
- [63] Microsoft Azure *Microsoft Azure IoT Edge*, <https://azure.microsoft.com/es-es/>, accessed: 2023
- [64] Node-RED *Node-RED*, <https://nodered.org/>, accessed: 2023
- [65] Homebridge *Homebridge*, <https://homebridge.io/>, accessed: 2023
- [66] Arrow *Guía Hardware de Arrow*, <https://www.arrow.com/en/ipe/download-area/som-guide>, accessed: 2023