# Objetivo

Documentar y realizar el proceso de creación de varios custom component de Home Assistant que trabajando conjuntamente implementen un controlador para una instalación fotovoltaica.

El desarrollo se realizará en Ubuntu con Python 3.12 en un entrono virtual en el que se instalará Home Assistant core. Desde Visual Studio Code se trabajará directamente sobre la carpeta custom\_components donde se crearán los componentes necesarios. A continuación se indican los componentes que se van a desarrollar. - pvpc\_sensor: dispone de los datos publicados desde la hora actual hasta la última información disponible de la web de ESIOS, para el PVPC, clasificados en franjas de 1hora. El valor del sensor será un array de pares fecha-hora y precios. En el arranque deberá rellenar el array lo más rápido posible, pero como máximo las llamadas a la API respetarán MIN\_TIME\_BETWEEN\_UPDATES. Debe disponer de propiedades para indicar el estado, y la fecha-hora en que se realizó la última toma de datos. - forecast\_solar\_sensor: dispone de los datos recogidos de forecast.solar para el pronostico de los watts que produce una instalación especificada en la configuración del sensor para 24 horas, clasificados en franjas de 15 minutos. El valor del sensor será un array de pares fecha-hora y watts. En el arranque deberá rellenar el array lo más rápido posible, pero como máximo las llamadas a la API respetarán MIN\_TIME\_BETWEEN\_UPDATES. Debe disponer de propiedades para indicar el estado, y la fecha-hora en que se realizó la última toma de datos. Debe disponer de flujo de configuración que permita introducir los valores que definen la instalacion (longitud, latitud, …) - inverter\_actuator: Se trata de un ModbusTcpClient que lee y escribe registros en el inversor. Como sensor ofrecerá multiples valores, entre ellos: currentSoC que indica el estado actual de carga de la batería; currentPower que indica la potencia consumida; y todos los parametros de configuración que el usuario introduzca durante el flujo de configuración, tienen que quedar accesibles para pv\_controller. Debe disponer de un método update\_inverter\_settings que pueda ser llamado por pv\_controller para establecer nuevas consignas en el inversor. Estas consignas serán grid\_setpoint, min\_allowed\_soc y force\_charge\_until. - pv\_controller: Rutina que se ejecuta periodicamente para hacer calculos con las informaciónes de pvpc\_sensor y forecast\_solar\_sensor y actaulizar las consignas de inverter\_actuator. El periodo con el que se ejecuta esta rutina deberá ser configurable mediante flujo de configurción en la UI

# Posibles enfoques para desarrollar en Home Assistant

Para introducirse en el desarrollo para Home Assistant (HA), es esencial tener claro qué tipo de desarrollo queremos realizar, ya que esto influirá directamente en el enfoque que tomemos y en las herramientas que utilizaremos. Los dos principales enfoques son:

1. **Modificación del código fuente de Home Assistant Core**: Este enfoque implica trabajar directamente sobre el código base de Home Assistant, lo que nos permite realizar contribuciones al proyecto oficial o personalizar aspectos profundos del sistema.
2. **Creación de Custom Components**: Este es el enfoque más habitual para quienes buscan ampliar las funcionalidades de Home Assistant mediante la creación de componentes personalizados que añadan nuevas integraciones, sensores o funcionalidades específicas.

### Recomendaciones iniciales

Independientemente del enfoque que elijas, es altamente recomendable trabajar en un entorno basado en Linux, como **Ubuntu**, ya que proporciona mayor flexibilidad y control sobre las dependencias y herramientas de desarrollo. Además, utilizar un editor de código avanzado como **Visual Studio Code** es crucial para mejorar la productividad, gracias a su integración con Git, la depuración, y sus múltiples extensiones para Python y YAML.

### Opciones para el entorno de desarrollo

Existen varias configuraciones posibles para el entorno de desarrollo de Home Assistant, dependiendo de tus preferencias y del tipo de proyecto que desees emprender:

1. **Contenedores Docker**:
   * Este enfoque es útil si deseas aislar tu entorno de desarrollo en un contenedor, lo que facilita la gestión de dependencias y versiones sin afectar el sistema principal. Es común cuando se trabaja directamente con el código fuente de Home Assistant, ya que puedes ejecutar fácilmente el proyecto en un contenedor que simula una instalación real de Home Assistant.
2. **Entornos virtuales de Python**:
   * Si prefieres evitar la complejidad de los contenedores, puedes optar por utilizar entornos virtuales de Python, que te permiten gestionar de manera aislada las dependencias de tu proyecto. Este método es flexible, fácil de configurar y proporciona un entorno limpio para el desarrollo de componentes personalizados o modificaciones en Home Assistant.

### Dos enfoques en entornos virtuales de Python

Si te decides por usar un entorno virtual de Python, existen dos formas comunes de configurar el entorno de trabajo:

1. **Clonando el repositorio oficial**:
   * Puedes empezar clonando el repositorio de Home Assistant desde GitHub:
   * git clone https://github.com/home-assistant/core.git
   * Esta opción es ideal si planeas trabajar directamente sobre el código fuente de Home Assistant Core, ya sea para enviar contribuciones al proyecto oficial o para probar cambios en una versión local. Con este enfoque, puedes tener control total sobre el código y aprovechar Dev Containers o herramientas avanzadas de desarrollo.
2. **Instalando Home Assistant en el entorno virtual**:
   * Otra opción es instalar Home Assistant directamente en el entorno virtual utilizando pip:
   * pip install homeassistant
   * Esta alternativa es más rápida y conveniente si solo necesitas desarrollar y probar Custom Components. Permite mantener un entorno de desarrollo limpio y separado del código fuente principal, lo que facilita la creación de componentes personalizados sin modificar el core de Home Assistant. Esto te permitirá centrarte en la creación de Custom Components o integraciones sin complicarte con el código fuente del sistema.

### Conclusión

Despues de experimentar con los distintos enfoques, para el desarrollo de Custom components se seguirá el enfoque de usar pip install homeassistant en un entorno virtual, que es más que suficiente para empezar.

# Metodología:

Para documentar y desarrollar estos custom components de Home Assistant, te recomiendo un enfoque estructurado que facilite tanto el desarrollo como la documentación clara del proceso. Aquí tienes un esquema sugerido:

### 1. **Entorno de Desarrollo**

* **Configuración del entorno**:
  + Asegúrate de tener Ubuntu configurado con Python 3.12 y un entorno virtual para aislar las dependencias de Home Assistant.
  + Instala Home Assistant Core y configura Visual Studio Code con las extensiones necesarias para trabajar con Python y YAML (para los archivos de configuración).
* **Control de versiones**:
  + Usa Git para seguir la evolución del proyecto y documentar los cambios. Asegúrate de realizar commits significativos que reflejen el progreso en cada componente.
  + Crea un repositorio separado para estos custom components.

### 2. **Componentes**

Para cada componente, estructura el desarrollo en fases:

#### **pvpc\_sensor**: - **Objetivo**: Leer el PVPC de la API de ESIOS y almacenar los datos en un array de pares fecha-hora/precios. - **Fases**: 1. Implementación del acceso a la API de ESIOS con manejo de errores y control de las actualizaciones mediante MIN\_TIME\_BETWEEN\_UPDATES. 2. Creación del sensor como un entity de Home Assistant, con las propiedades necesarias para el estado y la última actualización. 3. Documenta cómo se cargan los datos en el arranque y cómo se gestiona el array de datos.

#### **forecast\_solar\_sensor**: - **Objetivo**: Obtener los pronósticos de producción solar desde la API de forecast.solar. - **Fases**: 1. Implementación de la llamada a la API con los parámetros definidos en la configuración (latitud, longitud, potencia pico, etc.). 2. Procesar y almacenar los datos en un array de pares fecha-hora/watts. 3. Implementar el flujo de configuración para definir los parámetros de la instalación. 4. Asegurar el manejo eficiente de las actualizaciones y la estructura del array de pronóstico.

#### **inverter\_actuator**: - **Objetivo**: Interactuar con un inversor fotovoltaico usando Modbus TCP y proporcionar acceso a múltiples parámetros. - **Fases**: 1. Implementación del cliente ModbusTcpClient para leer y escribir registros. 2. Creación de los sensores para currentSoC, currentPower, y otros parámetros configurables. 3. Documenta la interacción con el inversor, incluyendo cómo se actualizan las consignas (grid\_setpoint, min\_allowed\_soc, force\_charge\_until). 4. Establecer un método que pueda ser llamado por el controlador fotovoltaico para ajustar los parámetros en función de los cálculos realizados.

#### **pv\_controller**: - **Objetivo**: Ejecutar una rutina periódica que ajuste las consignas del inversor en base a los datos de los otros sensores. - **Fases**: 1. Desarrollo de los cálculos necesarios para combinar los datos de pvpc\_sensor y forecast\_solar\_sensor. 2. Integrar las consignas generadas y ajustarlas a través de inverter\_actuator. 3. Implementar un flujo de configuración en la UI de Home Assistant para definir el periodo de ejecución de la rutina. 4. Testear el controlador con escenarios prácticos y diferentes configuraciones.

### 3. **Documentación**

Asegúrate de documentar cada fase del desarrollo: - **Código**: Documenta el código de cada componente explicando cómo funciona cada bloque y la lógica detrás de las decisiones. - **Configuración**: Proporciona instrucciones claras para la configuración de cada sensor y actuador en Home Assistant. - **Dependencias**: Anota las dependencias y versiones requeridas. - **Pruebas**: Incluye una sección de pruebas con ejemplos prácticos de cómo usar el controlador, así como las diferentes configuraciones posibles.

### 4. **Optimización y Testeo**

* **Testeo en Home Assistant**: Cada componente debe ser probado en Home Assistant para asegurarse de que se integra correctamente y funciona según lo esperado.
* **Manejo de errores**: Asegúrate de tener un buen manejo de excepciones, especialmente para las conexiones con las APIs y el inversor.
* **Optimización de llamadas a la API**: Minimiza las llamadas a las APIs externas respetando los tiempos mínimos entre actualizaciones.

### 5. **Despliegue**

* **Distribución**: Considera empaquetar estos custom components de forma modular para facilitar su uso por otros usuarios de Home Assistant.
* **Mantenimiento**: Documenta cómo actualizar los componentes en el futuro o cómo realizar ajustes.

En el siguiente documento se encuentran los pasos de configuración del entorno para Visual Studio Code en Ubuntu [vs\_code\_first\_steps](Docs/vs_code_first_steps.md)