EXPLICACION DE LOS CODIGOS DE **SENALODOO**

CODIGOS DE ARDUINO (MENSAJERIA DE ARDUINO):

1-. Mensajeriaodoo3:

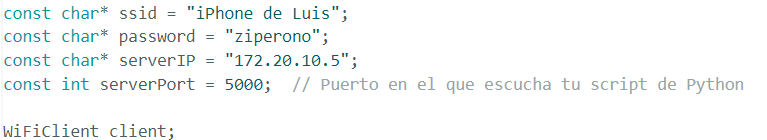




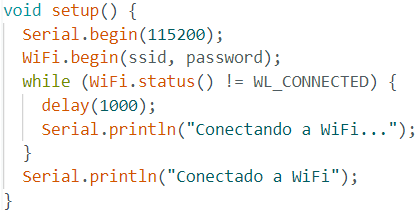
Este código es para un microcontrolador ESP32 (o ESP8266) que se conecta a WiFi y envía diferentes tipos de "pulsos" a un servidor Python. Vamos a desglosar las funciones y su funcionamiento:



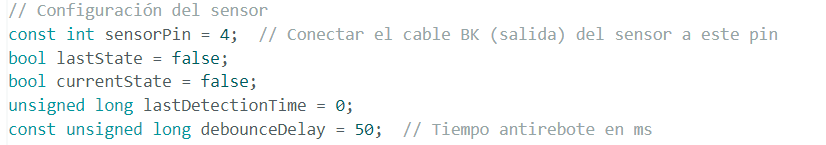
* **WiFi.h**: Librería para manejar la conexión WiFi en ESP32.



* Configuración de red: SSID, contraseña, IP del servidor y puerto.
* **WiFiClient**: Objeto para manejar conexiones cliente TCP.

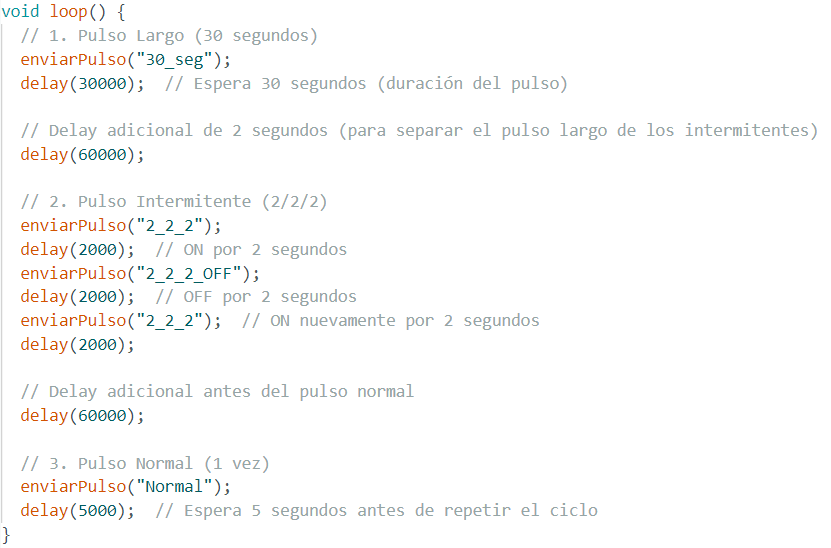


* **Serial.begin(115200)**: Inicia comunicación serial para depuración.
* **WiFi.begin(ssid, password)**: Intenta conectarse a la red WiFi.
* **WiFi.status()**: Verifica el estado de la conexión (WL\_CONNECTED = conexión exitosa).

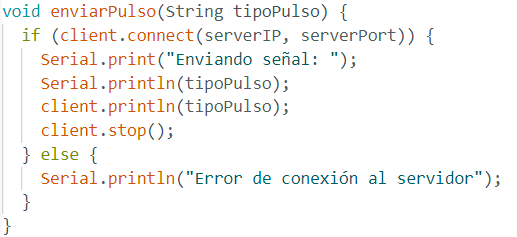




* El sensor es de tipo PNP (entrega voltaje positivo cuando detecta).



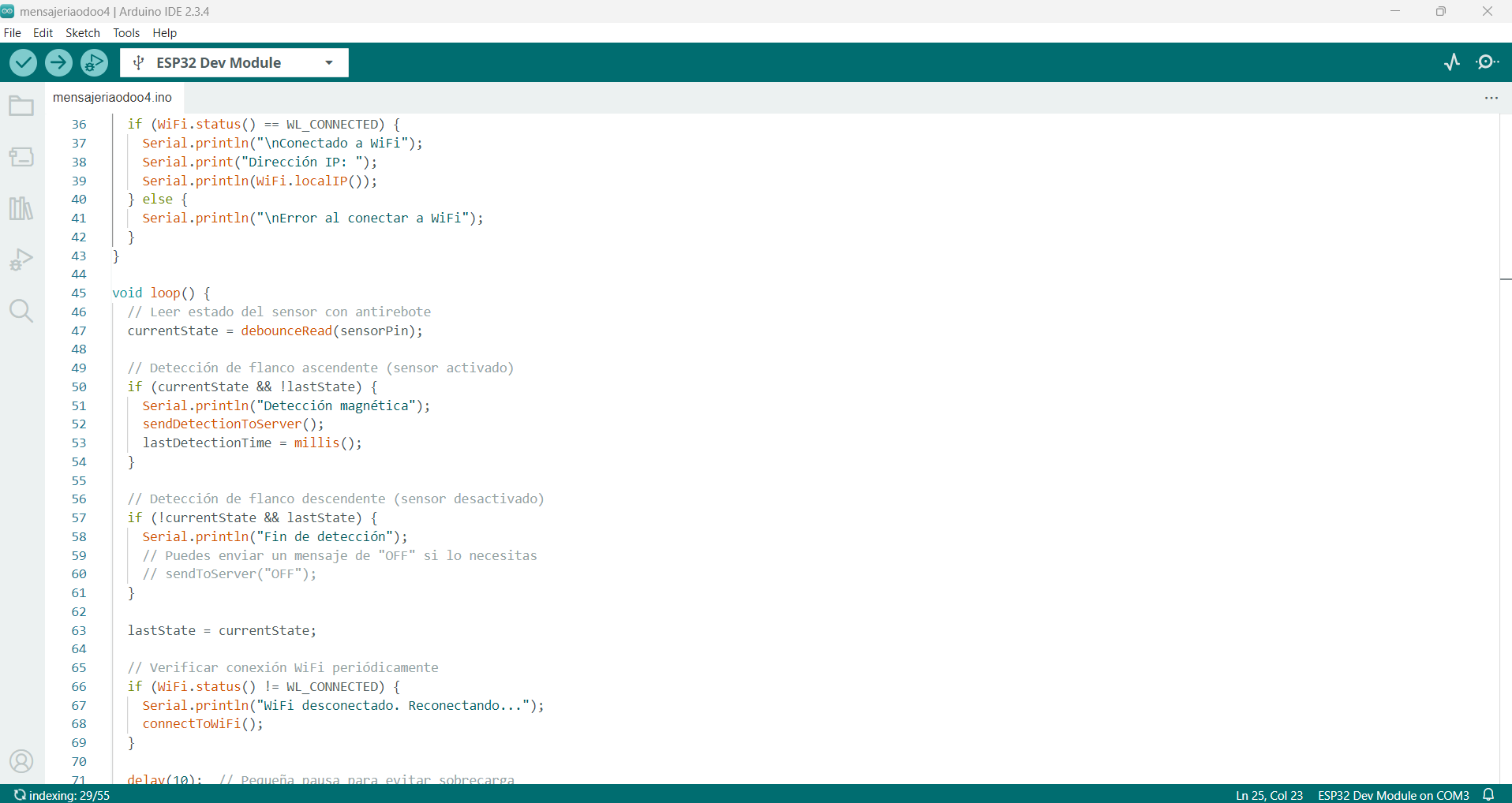
* Implementa un ciclo con 3 patrones de pulsos diferentes con retardos entre ellos.



* **client.connect()**: Establece conexión TCP con el servidor.
* **client.println()**: Envía el mensaje al servidor.
* **client.stop()**: Cierra la conexión después de enviar.

2-. Mensajeriaodoo4:



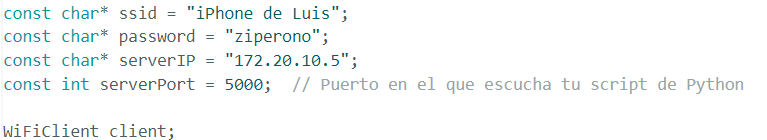




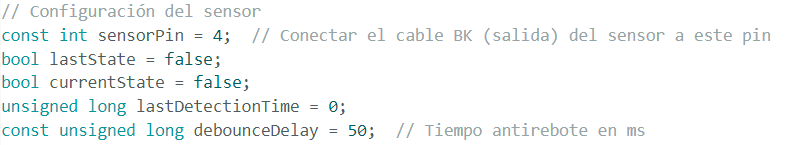
Este código es para un microcontrolador ESP32/ESP8266 que monitorea un sensor magnético y envía notificaciones a un servidor de Python cuando detecta cambios. Aquí está el desglose detallado:



* **WiFi.h**: Librería para manejar la conexión WiFi en ESP32.



* Configuración de red: SSID, contraseña, IP del servidor y puerto.
* **WiFiClient**: Objeto para manejar conexiones cliente TCP.



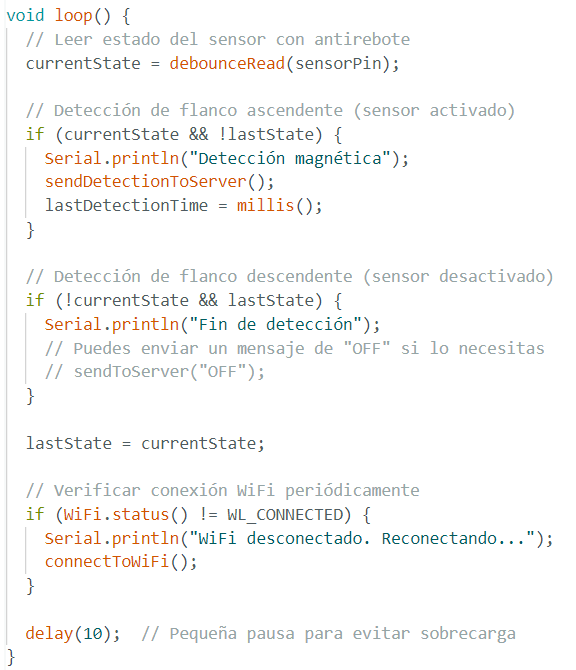
* El sensor es de tipo PNP (entrega voltaje positivo cuando detecta).



* Inicia comunicación serial y configura el pin del sensor con resistencia pull-down.



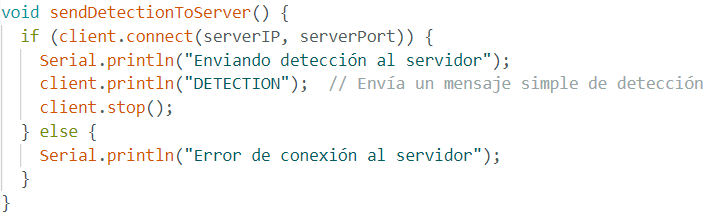
* Maneja la conexión WiFi con timeout de 10 segundos.



* Monitorea continuamente el sensor y gestiona la conexión WiFi.



* Implementa antirebote (debounce) para evitar falsas detecciones por ruido eléctrico.



* Establece conexión TCP, envía mensaje y cierra la conexión.

3-. Mensajeriaodoo5:





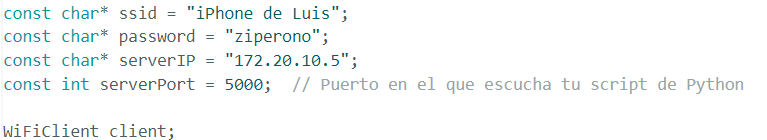




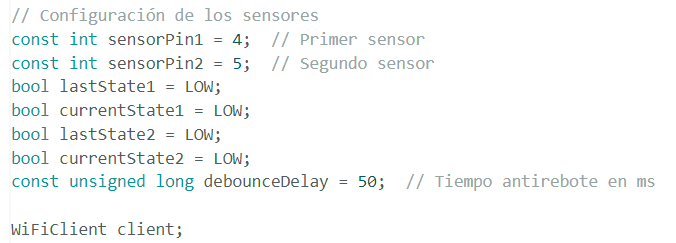
Este código es para un microcontrolador ESP32/ESP8266 que monitorea un sensor magnético y envía notificaciones a un servidor de Python cuando detecta cambios. Aquí está el desglose detallado:



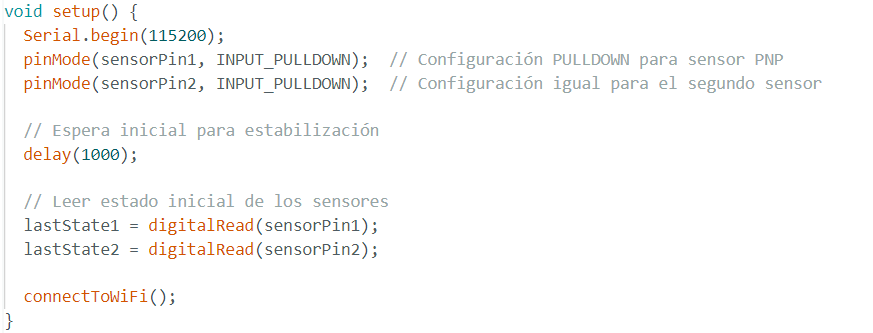
* **WiFi.h**: Librería para manejar la conexión WiFi en ESP32.



* Configuración de red: SSID, contraseña, IP del servidor y puerto.
* **WiFiClient**: Objeto para manejar conexiones cliente TCP.

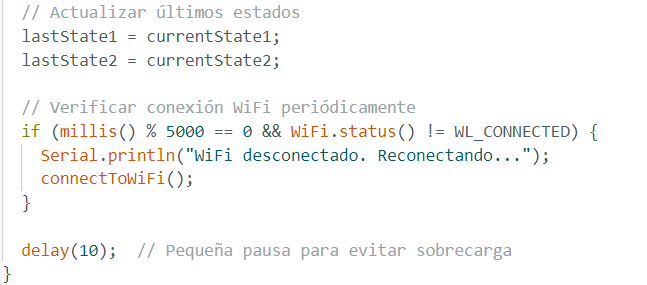


* Soporta **dos sensores PNP** (entregan +24V cuando activos)
* Mantiene **estados separados** para cada sensor
* Mismo tiempo de antirebote (50ms) para ambos

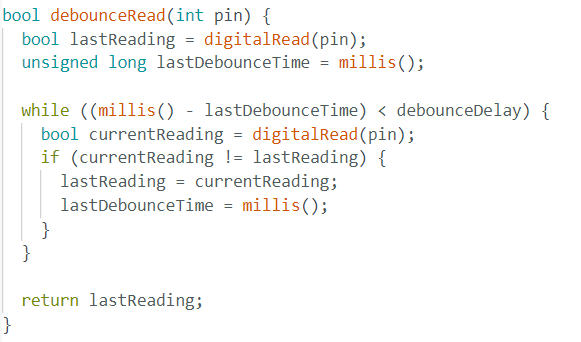


* Inicializa **ambos sensores** con resistencia pull-down
* Estabilización de 1 segundo al inicio

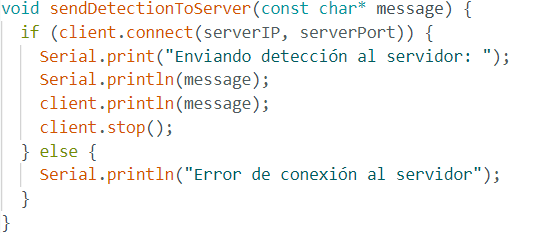




* **Optimización clave**: Solo aplica debounce cuando detecta cambio
* Envía mensajes diferenciados (DETECTION1/DETECTION2)
* Verificación WiFi periódica usando millis()



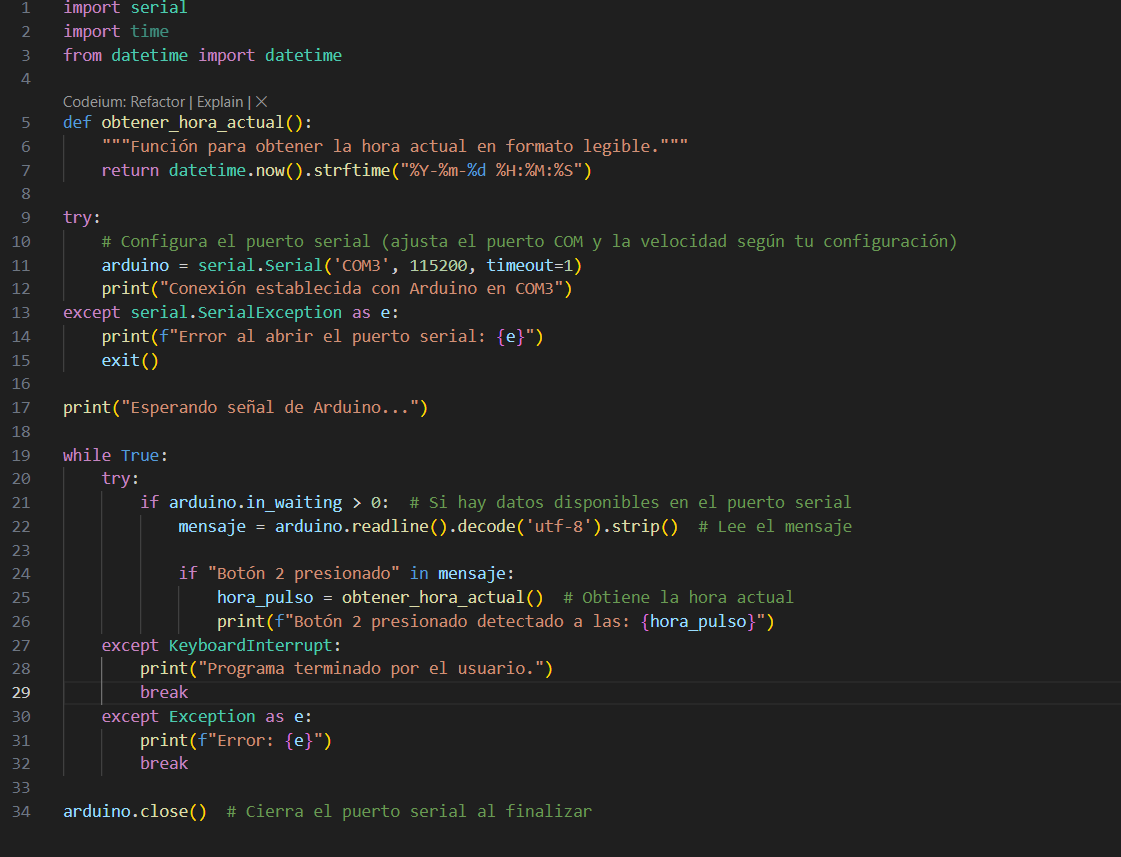
* **Función reutilizable** para ambos sensores
* Misma implementación que en el ejemplo anterior



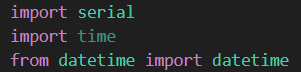
* Versión mejorada que acepta **mensajes personalizados**
* Diferenciación clara entre sensores

PUERTOS DE ODOO VERSIONES:

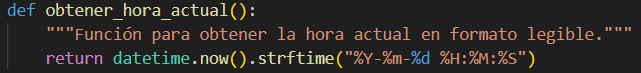
1-. Puerto:



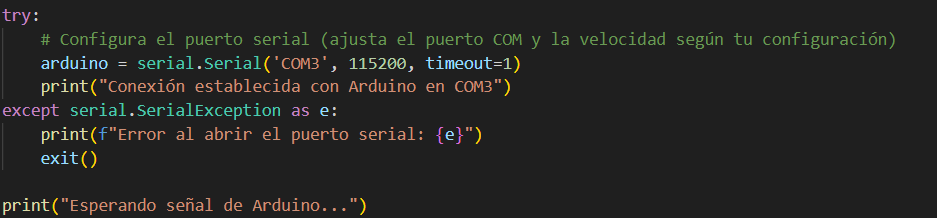
Este script Python se comunica con un Arduino a través del puerto serial para detectar pulsaciones de botones y registrar el momento exacto en que ocurren. (Un código base que todavía no lograba la conexión a Odoo):



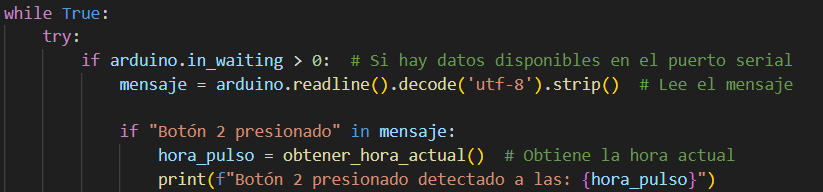
* **serial**: Para comunicación serial con Arduino
* **time**: Para manejo de tiempos (aunque no se usa directamente)
* **datetime**: Para obtener marcas de tiempo precisas



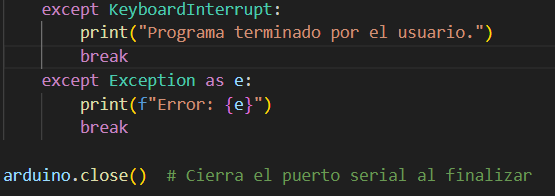
* Devuelve la fecha y hora actual en formato AAAA-MM-DD HH:MM:SS
* Se usa para registrar el momento exacto de las pulsaciones



* Intenta conectar con Arduino en COM3 a 115200 baudios
* timeout=1 establece un tiempo máximo de espera para operaciones de lectura
* Maneja errores si no puede establecer la conexión

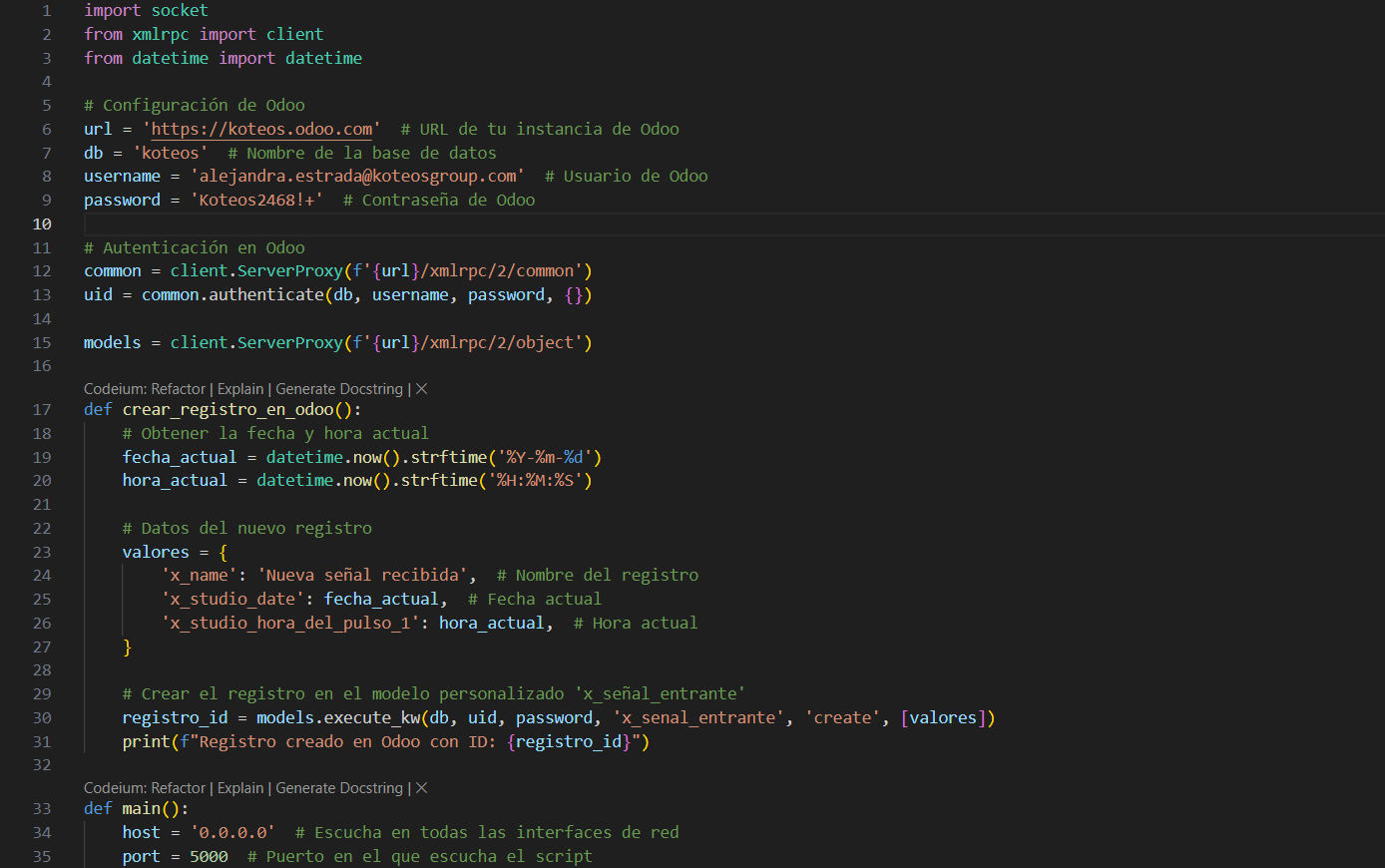


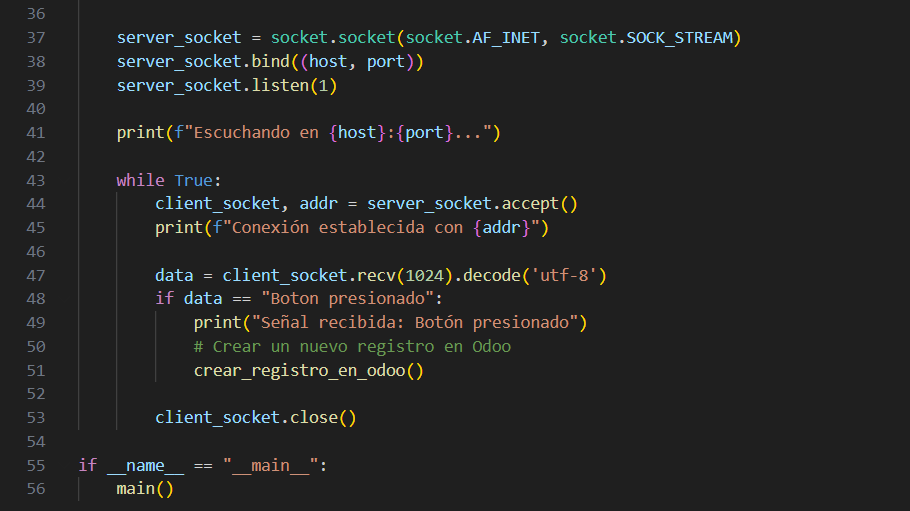
* **arduino.in\_waiting**: Verifica si hay datos disponibles en el buffer serial
* **readline()**: Lee una línea completa del puerto serial
* **decode('utf-8').strip()**: Convierte bytes a string y elimina espacios/retornos de carro
* Cuando detecta "Botón 2 presionado", registra la hora exacta



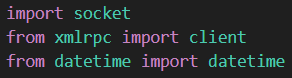
* **KeyboardInterrupt**: Captura Ctrl+C para una terminación elegante
* **Exception general**: Captura cualquier otro error inesperado
* Libera el puerto serial cuando termina el programa

2-. Puertoodoo:

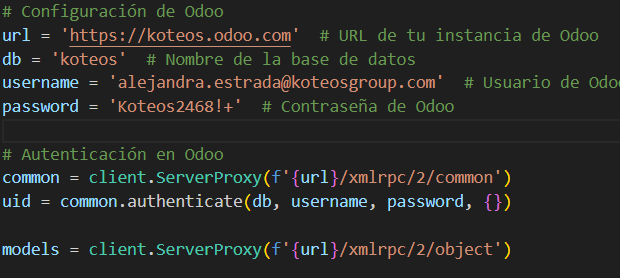




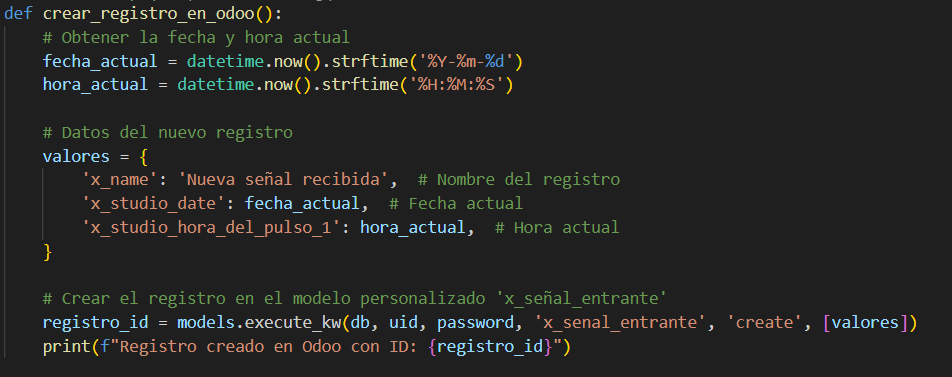
Este script implementa un servidor TCP que recibe señales y crea registros en Odoo a través de su API XML-RPC. Vamos a analizarlo en detalle:



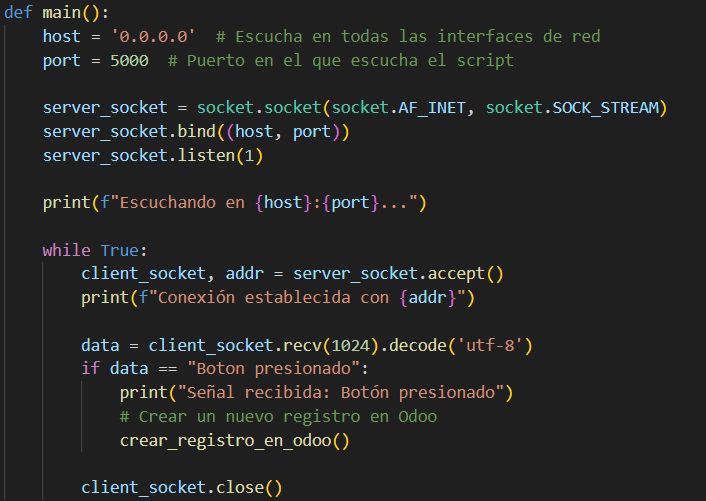
* **url**: Endpoint de la instancia de Odoo
* **db**: Nombre de la base de datos
* **username/password**: Credenciales de autenticación



* Establece conexión con los endpoints XML-RPC de Odoo
* authenticate() obtiene el UID del usuario
* models permite interactuar con los modelos de Odoo



* **Obtiene fecha/hora actual**: Formateadas para Odoo
* **Estructura de datos**: Mapea a campos personalizados en Odoo (x\_studio\_\*)
* **execute\_kw**: Método clave que crea el registro en el modelo personalizado 'x\_senal\_entrante'

****

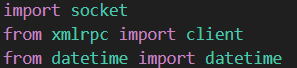
* **Servidor TCP**: Escucha en puerto 5000
* **Flujo**: Acepta conexiones → Recibe datos → Si es "Boton presionado" → Crea registro en Odoo
* **0.0.0.0**: Acepta conexiones de cualquier interfaz de red

3-. Puertoodoo2:

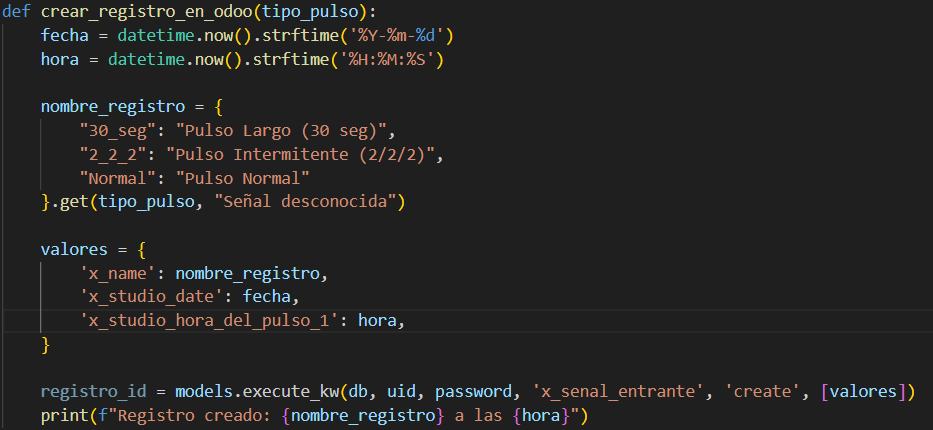
Este script es una versión mejorada del anterior que recibe diferentes tipos de pulsos desde un dispositivo ESP32/ESP8266 y los registra en Odoo con descripciones específicas.

**Mejoras Principales**

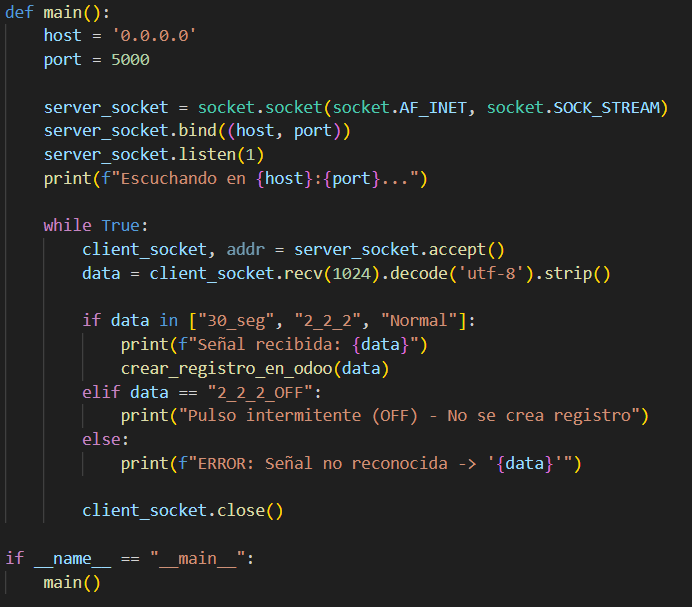
1. **Manejo de múltiples tipos de pulsos**:
   * Pulso largo (30 seg)
   * Pulso intermitente (2/2/2)
   * Pulso normal
   * Estado OFF para pulsos intermitentes
2. **Registro descriptivo en Odoo**:
   * Asigna nombres comprensibles a cada tipo de pulso
   * Registra la hora exacta de recepción



* Misma configuración de autenticación que en la versión anterior

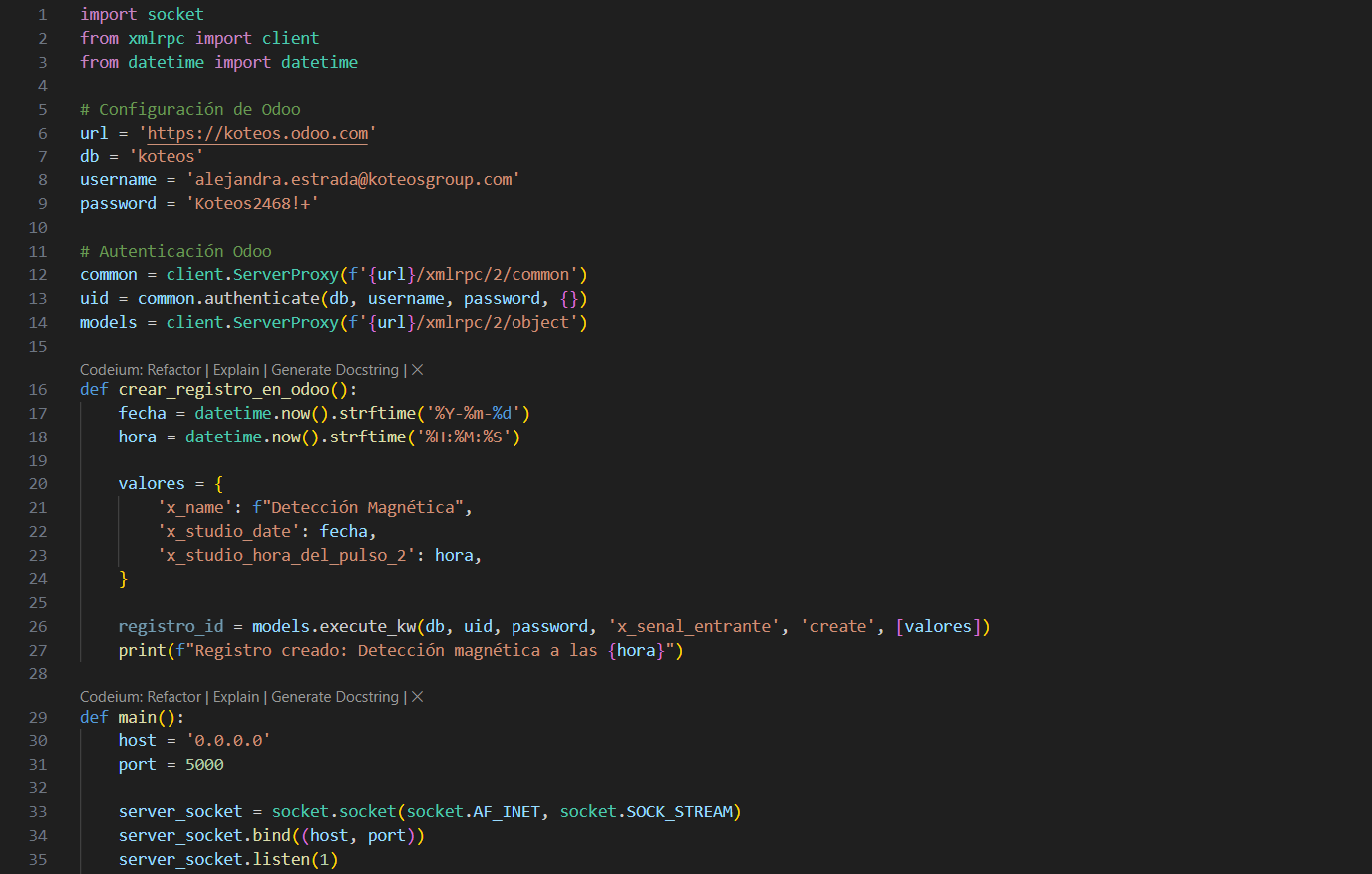


* **Novedad**: Usa un diccionario para mapear códigos de pulso a nombres descriptivos
* **Flexibilidad**: Maneja señales desconocidas con un texto por defecto
* **Eficiencia**: Obtiene fecha y hora solo cuando es necesario



* **Validación mejorada**: Filtra explícitamente los tipos de pulsos aceptados
* **Manejo especial**: Para el estado OFF del pulso intermitente
* **Registro de errores**: Muestra señales no reconocidas

4-. Puertoodoo3:

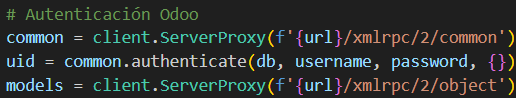




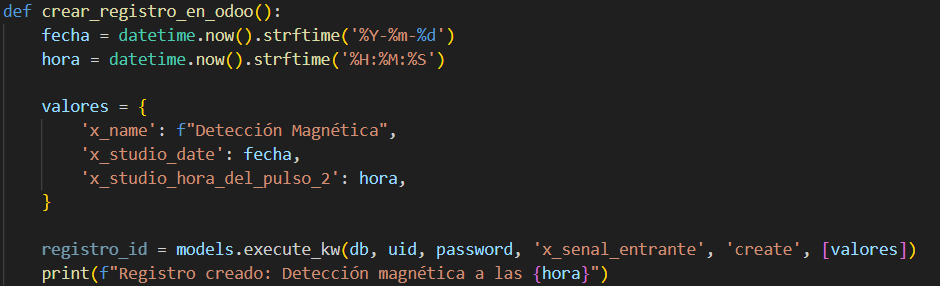
Este script implementa un servidor TCP que recibe señales de detección magnética desde un dispositivo ESP32/ESP8266 y registra estos eventos en Odoo.



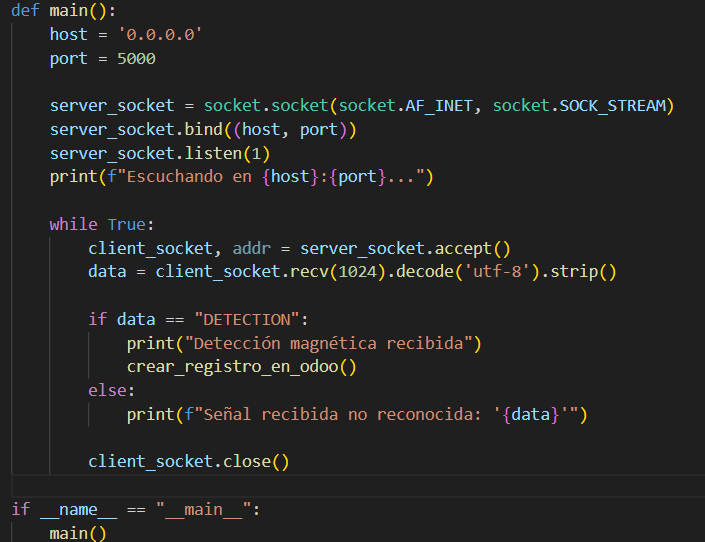
* **url**: Endpoint de la instancia Odoo
* **db**: Nombre de la base de datos
* **username/password**: Credenciales de acceso



* Establece conexión con los servicios XML-RPC de Odoo
* authenticate() obtiene el user ID para operaciones posteriores
* models permite interactuar con los modelos de la base de datos

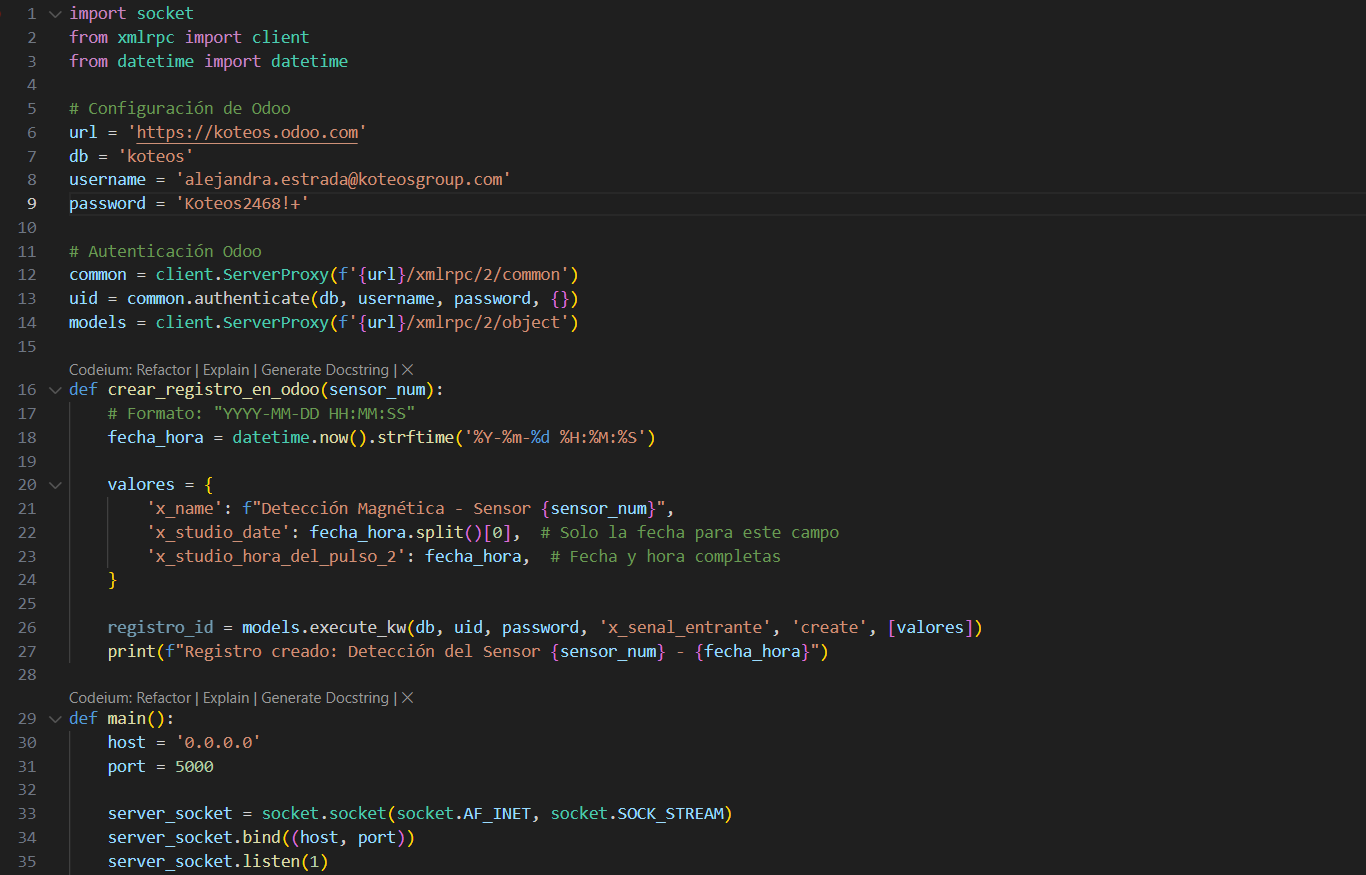


* **Captura fecha/hora actual**: Con formato para Odoo
* **Estructura de datos**:
  + x\_name: Descripción del evento
  + x\_studio\_date: Fecha de detección
  + x\_studio\_hora\_del\_pulso\_2: Hora exacta del evento
* **execute\_kw**: Crea el registro en el modelo personalizado 'x\_senal\_entrante'



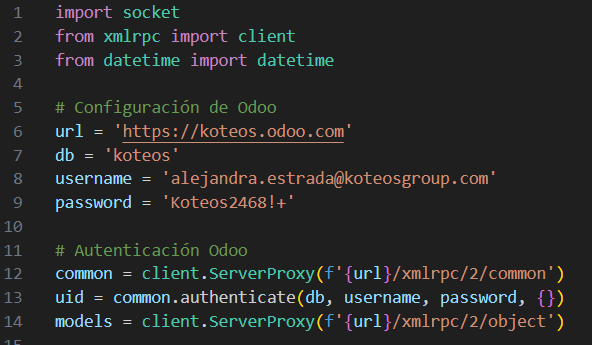
* **Servidor TCP**: Escucha en puerto 5000
* **Flujo de operación**:
  1. Acepta conexiones entrantes
  2. Recibe y decodifica mensajes
  3. Valida si es señal "DETECTION"
  4. Crea registro en Odoo o reporta error
  5. Cierra conexión

5-. Puertoodoo4:

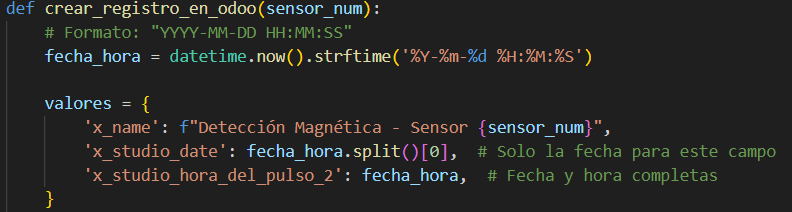




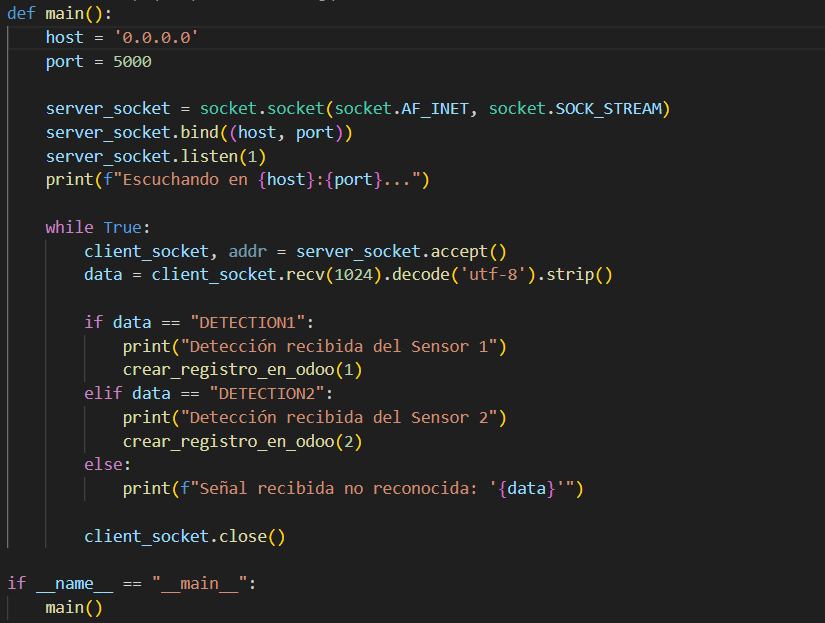
Este script es una versión avanzada que maneja **dos sensores magnéticos independientes**, registrando sus detecciones en Odoo con información diferenciada.



* **url**: Endpoint de la instancia Odoo
* **db**: Nombre de la base de datos
* **username/password**: Credenciales de acceso
* Establece conexión con los servicios XML-RPC de Odoo
* authenticate() obtiene el user ID para operaciones posteriores
* models permite interactuar con los modelos de la base de datos



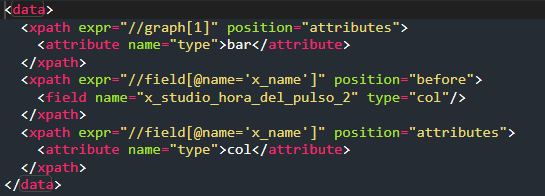
* **Registro diferenciado**: Identifica claramente qué sensor activó la detección
* **Formato temporal mejorado**: Usa marca temporal completa en un campo y solo fecha en otro



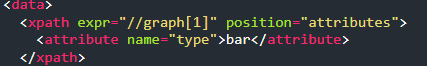
* **Recepción diferenciada**: Detecta señales de dos sensores distintos
* **Lógica clara**: Asigna número de sensor correspondiente

XML dentro de Odoo:

1-. GRAFICACION:



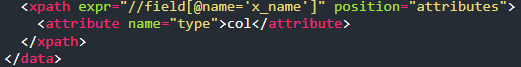
Este archivo XML configura una vista gráfica en Odoo para visualizar los datos de detección magnética que hemos estado registrando desde los scripts Python.



* **graph[1]**: Selecciona el primer elemento gráfico encontrado
* **type="bar"**: Establece el tipo de gráfica como **barras verticales**
* Alternativas comunes: pie (tarta), line (líneas), bar (barras)



* **position="before"**: Inserta el campo antes del campo x\_name
* **x\_studio\_hora\_del\_pulso\_2**: Muestra la marca temporal completa
* **type="col"**: Indica que será una columna en la gráfica



* Convierte el campo x\_name en una columna de la gráfica
* Mostrará los valores "Detección Magnética - Sensor 1" y "Detección Magnética - Sensor 2"