

**Manual técnico: Implementación de un sistema de monitoreo IoT para invernadero  
hidropónico utilizando Arduino XIAO ESP32-C3 y ESP32-CAM vía MQTT**

Daniel Felipe Ruiz Lizcano - 20222207514

Luisa María José Coronel - 20222208436

Universidad Surcolombiana

Facultad de Ingeniería, programa de Ingeniería de Software

Ingeniero: Juan Antonio Castro Silva

Neiva, Colombia

30 de Mayo de 2025

## Tabla de contenido

Objetivos.....	3
Objetivo General.....	3
Objetivos Específicos.....	3
Introducción.....	4
1. Instalación del entorno de desarrollo (Arduino IDE).....	5
1.1 Descarga e instalación del Arduino IDE.....	5
1.2 Instalación de las placas ESP32 desde el Gestor de tarjetas.....	6
2. Instalación de la placa Sseed Studio XIAO ESP32-C3.....	7
2.1 Añadir soporte para la placa Sseed Studio XIAO ESP32-C3.....	7
2.2 Seleccionar la placa correcta.....	7
3. Instalación de librerías necesarias.....	9
3.1 Acceder al Gestor de librerías.....	9
3.2 Librerías a instalar.....	9
3.3 Instalación de librerías desde el Gestor.....	10
4. Modificación del archivo PubSubClient.h.....	11
4.1 Ubicación del archivo PubSubClient.h.....	11
4.2 Instrucciones para la modificación.....	11
4.3 Justificación del cambio.....	11
4.4 Verificación posterior a la modificación.....	12
5. Conexión física del circuito.....	13
5.1 Conexiones del Arduino XIAO ESP32-C3.....	13
5.2 Conexiones de la ESP32-CAM (AI Thinker).....	14
6. Configuración de red WiFi y dirección del broker MQTT.....	17
6.1 Parámetros en el código del Arduino XIAO ESP32-C3.....	17
6.2 Parámetros en el código de la ESP32-CAM.....	18
6.3 Prueba de conectividad.....	18
7. Subida del código.....	19
7.1 Subida del código al Arduino XIAO ESP32-C3.....	19
7.2 Subida del código a la ESP32-CAM (AI Thinker).....	20
8. Verificación de funcionamiento.....	22
8.1 Verificación en el Arduino XIAO ESP32-C3.....	22
8.2 Verificación en la ESP32-CAM AI Thinker.....	23
8.3 Validación desde el broker.....	24
9. Conclusiones y recomendaciones finales.....	24
9.1 Conclusiones clave.....	24
9.2 Recomendaciones técnicas.....	25
9.3 Escalabilidad y mejoras futuras.....	26
Referencias.....	27

## **Objetivos**

### **Objetivo General**

Diseñar e implementar un sistema de monitoreo y control ambiental para un invernadero hidropónico inteligente, basado en tecnologías IoT mediante el uso de microcontroladores Arduino XIAO ESP32-C3 y ESP32-CAM, que permita la recolección de datos climáticos, el control automático de dispositivos actuadores, y la transmisión de imágenes en tiempo real a través del protocolo MQTT hacia un broker local.

### **Objetivos Específicos**

1. Configurar el entorno de desarrollo en Arduino IDE para la programación de los microcontroladores utilizados, asegurando la compatibilidad con las placas ESP32-C3 y ESP32-CAM.
2. Implementar la adquisición de datos ambientales mediante el sensor DHT22 conectado al Arduino XIAO ESP32-C3, para medir temperatura y humedad relativa del invernadero.
3. Programar el control automático de actuadores (ventilador y sistema de iluminación) en función de umbrales térmicos definidos, utilizando salidas digitales del microcontrolador.
4. Integrar la captura de imágenes del cultivo a través del módulo ESP32-CAM, codificando los datos en formato Base64 para su transmisión vía MQTT.
5. Optimizar la comunicación entre dispositivos mediante el protocolo MQTT, configurando los parámetros de red y modificando los límites del paquete de datos en la librería PubSubClient.
6. Diseñar la arquitectura física del sistema, detallando la conexión eléctrica de sensores, actuadores y módulos de comunicación para garantizar un funcionamiento estable y seguro.
7. Verificar el funcionamiento completo del sistema mediante pruebas de conectividad WiFi, publicación/recepción de mensajes MQTT y visualización de datos e imágenes en tiempo real.

## Introducción

En la actualidad, la agricultura de precisión se ha visto potenciada por la integración de tecnologías IoT (Internet de las Cosas), permitiendo la automatización, monitoreo y control remoto de variables críticas en ambientes agrícolas controlados, como los invernaderos hidropónicos. Este manual tiene como objetivo guiar paso a paso el desarrollo de un sistema de monitoreo ambiental para un invernadero, utilizando dos microcontroladores de bajo consumo y alto rendimiento: el Arduino XIAO ESP32-C3 y la ESP32-CAM (modelo AI Thinker).

El proyecto consiste en medir y controlar variables ambientales relevantes como temperatura, humedad, ventilación e iluminación, además de capturar imágenes periódicas del cultivo, enviando todos los datos a través del protocolo MQTT hacia un broker local (Mosquitto). El uso de dos dispositivos diferenciados permite una distribución modular de tareas:

El Arduino XIAO ESP32-C3 está encargado de recolectar datos desde sensores como el DHT22, y controlar actuadores como un relé para iluminación y un ventilador.

La ESP32-CAM, por su parte, captura imágenes del invernadero y las envía codificadas en Base64 mediante MQTT al mismo broker.

A lo largo del documento, se describirán todos los pasos técnicos necesarios para instalar el entorno de desarrollo, configurar las placas, realizar las conexiones físicas, cargar los códigos y verificar el funcionamiento correcto del sistema. Además, se brindarán recomendaciones prácticas para la resolución de errores comunes y la optimización del funcionamiento en red.

Este manual está diseñado para estudiantes, técnicos e ingenieros que busquen implementar soluciones IoT funcionales en contextos reales, combinando programación embebida, electrónica y redes de comunicación.

## 1. Instalación del entorno de desarrollo (Arduino IDE)

Para programar y cargar los códigos en los microcontroladores utilizados en este proyecto (Arduino XIAO ESP32-C3 y ESP32-CAM AI Thinker), se empleará el Arduino IDE, una plataforma de desarrollo de código abierto ampliamente utilizada en el ámbito de la electrónica y la programación embebida.

### 1.1 Descarga e instalación del Arduino IDE


1. Ingresar al sitio web oficial de Arduino:  
<https://www.arduino.cc/en/software>
2. Seleccionar la versión correspondiente al sistema operativo del usuario (Windows, macOS o Linux).
3. Descargar el instalador y ejecutar el archivo para iniciar el proceso de instalación.
4. Durante la instalación, aceptar los términos y condiciones, y permitir la instalación de los drivers necesarios para el reconocimiento de dispositivos.

Nota: Se recomienda instalar la versión más reciente para asegurar compatibilidad con placas modernas como las basadas en el chip ESP32.

### Figura 1

*Instalación del entorno Arduino IDE desde el sitio oficial (Arduino, s.f.).*

## Downloads



### Arduino IDE 2.3.6

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the [Arduino IDE 2.0 documentation](#).

Nightly builds with the latest bugfixes are available through the section below.

#### DOWNLOAD OPTIONS

**Windows** Win 10 and newer, 64 bits  
**Windows** MSI installer  
**Windows** ZIP file

**Linux** AppImage 64 bits (X86-64)  
**Linux** ZIP file 64 bits (X86-64)

**macOS** Intel, 10.15: "Catalina" or newer, 64 bits  
**macOS** Apple Silicon, 11: "Big Sur" or newer, 64 bits

[Release Notes](#)

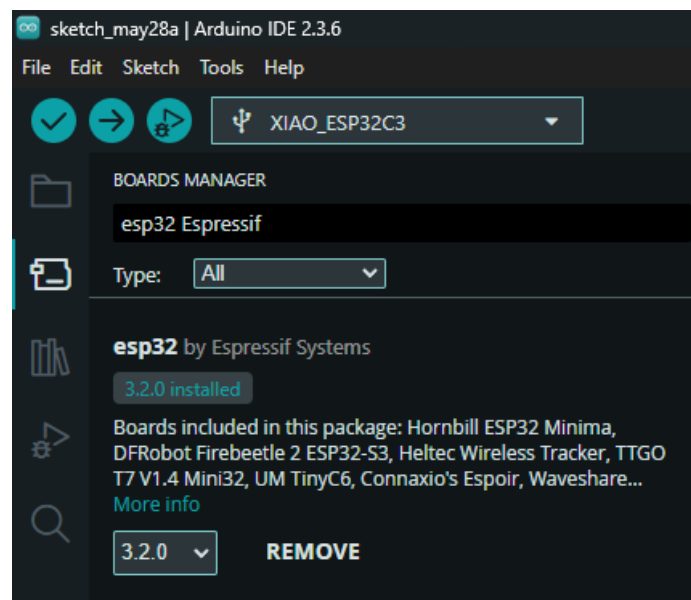
## 1.2 Instalación de las placas ESP32 desde el Gestor de tarjetas

1. Ir al menú lateral izquierdo, selecciona “Board Manager”
2. En la barra de búsqueda, escribir
3. Seleccionar el paquete esp32 by Espressif Systems y hacer clic en “Instalar”.
4. Esperar a que finalice la instalación.

Este paso instalará soporte para una amplia gama de placas ESP32, incluyendo la ESP32-CAM y la Sseed XIAO ESP32-C3.

### Figura 2

*Instalación del paquete de placas ESP32 desde el Gestor de tarjetas (Arduino, s.f.).*



## **2. Instalación de la placa Seeed Studio XIAO ESP32-C3**

El Seeed Studio XIAO ESP32-C3 es un microcontrolador compacto basado en el chip ESP32-C3 de Espressif, compatible con el ecosistema de Arduino. Para poder programarlo desde el Arduino IDE, se deben seguir ciertos pasos específicos para su instalación y selección dentro del entorno.

### **2.1 Añadir soporte para la placa Seeed Studio XIAO ESP32-C3**

Una vez instalado el soporte general para placas ESP32 (ver Sección 1), el Arduino IDE ya incluye soporte para el modelo XIAO ESP32-C3 dentro del mismo paquete de Espressif.

Pasos:

1. Abrir el Arduino IDE.
2. Ir a Herramientas > Placa > Gestor de tarjetas.
3. En el buscador, escribir: esp32
4. Confirmar que el paquete esp32 by Espressif Systems esté instalado.

En versiones recientes, el XIAO ESP32-C3 ya aparece como una de las opciones incluidas por defecto al instalar el paquete de Espressif.

### **2.2 Seleccionar la placa correcta**

Para compilar y subir el código correctamente, es fundamental seleccionar el modelo de placa exacto en el IDE.

Pasos:

1. Ir a Herramientas > Placa.
2. Buscar y seleccionar:

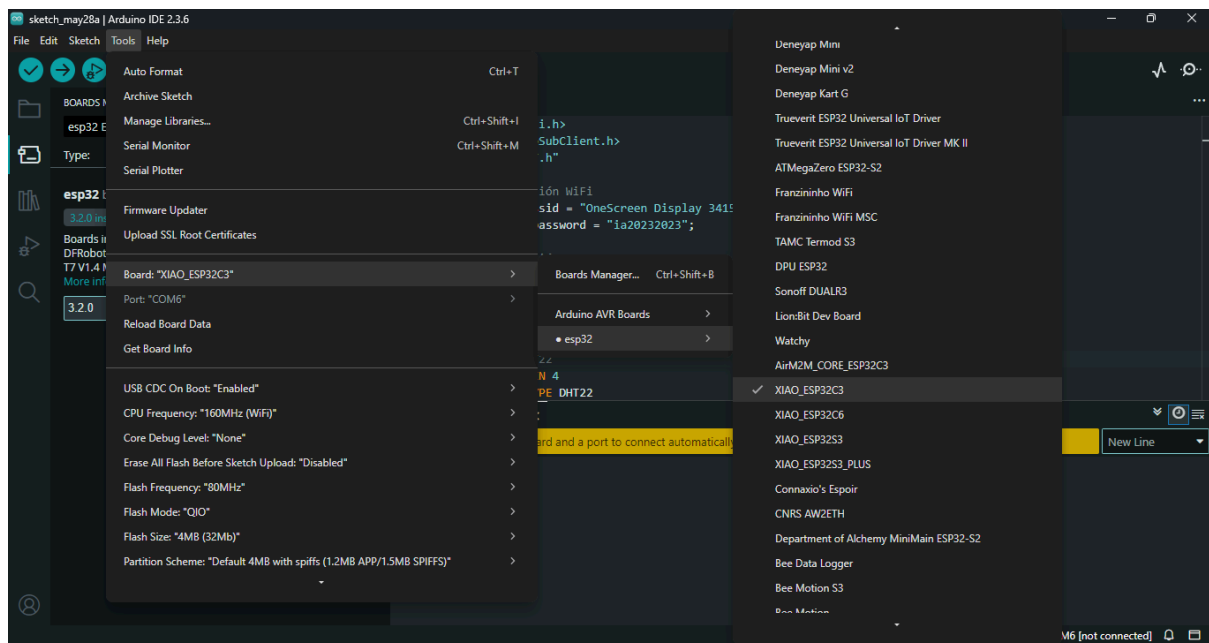
"XIAO ESP32C3"

### 3. Configurar las siguientes opciones recomendadas en el mismo menú de Herramientas:

- Velocidad de carga (Upload Speed): 115200
- Flash Mode: QIO
- Partition Scheme: Default 4MB with spiiffs (si está disponible)
- Core Debug Level: Ninguno (None)
- Port (Puerto): Se seleccionará automáticamente al conectar el dispositivo por USB

**Figura 3**

*Selección de la placa Seeed Studio XIAO ESP32-C3 en el Arduino IDE (Arduino, s.f.).*





### 3. Instalación de librerías necesarias

Para que los programas desarrollados puedan compilarse y funcionar correctamente en el Arduino IDE, es necesario instalar varias librerías externas. Estas permiten interactuar con los sensores, manejar la conexión MQTT y codificar imágenes en Base64.

#### 3.1 Acceder al Gestor de librerías

1. Abrir el Arduino IDE.
2. Ir al menú:  
     Programa > Incluir Librería > Administrar bibliotecas...  
     (o directamente desde el ícono de bibliotecas en la barra lateral).
3. Se abrirá el Gestor de Librerías, donde se pueden buscar e instalar las necesarias.

#### 3.2 Librerías a instalar

A continuación, se enumeran las librerías requeridas, junto con su función y el nombre exacto que debe usarse en el buscador del gestor:

Nombre en el gestor	Uso en el proyecto
<b>PubSubClient by Nick O'Leary</b>	Comunicación MQTT (publicar y suscribirse a tópicos)
<b>DHT sensor library by Adafruit</b>	Lectura del sensor DHT22 de temperatura y humedad
<b>Adafruit Unified Sensor</b>	Soporte común para sensores Adafruit

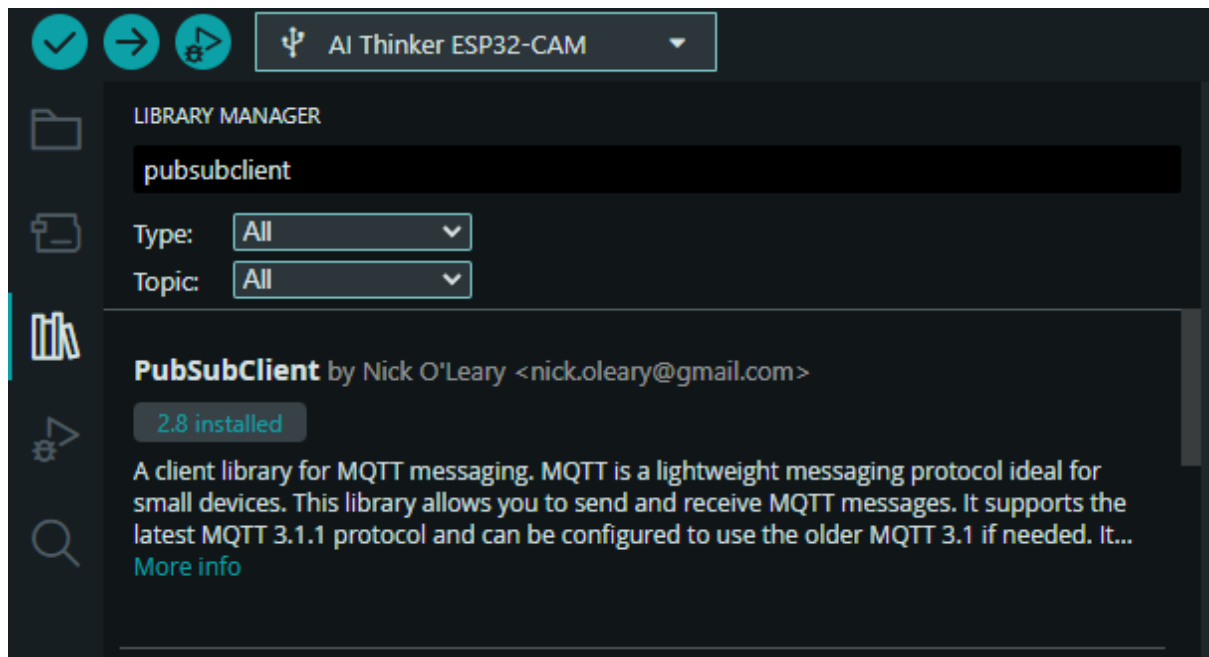
Nota: La librería base64.h no se encuentra en el gestor y debe crearse manualmente.  
 Ver el archivo completo proporcionado (en la Sección 3.4 de este manual).

### 3.3 Instalación de librerías desde el Gestor

1. En el Gestor de Librerías, usar el campo de búsqueda para localizar cada una de las librerías anteriores.
2. Hacer clic en “Instalar” cuando se muestre el resultado correcto.
3. Verificar que estén instaladas las versiones más recientes compatibles.

**Figura 4**

*Instalación de la librería "PubSubClient" desde el Gestor de Librerías (Arduino, s.f.).*



## 4. Modificación del archivo PubSubClient.h

La librería PubSubClient está diseñada para mensajes MQTT de tamaño reducido, ideales para sensores. Sin embargo, en este proyecto, la ESP32-CAM captura imágenes JPEG que se codifican en Base64, generando cadenas largas de texto. Para permitir su transmisión, es necesario aumentar el tamaño máximo del paquete MQTT permitido por la librería.

### 4.1 Ubicación del archivo PubSubClient.h

Este archivo se encuentra dentro de la carpeta de librerías del entorno de Arduino. La ruta típica en sistemas Windows es:

“Documentos/Arduino/libraries/PubSubClient/src/PubSubClient.h”

En macOS o Linux, la ruta puede variar, pero siempre estará dentro de la carpeta libraries/PubSubClient/src/.

### 4.2 Instrucciones para la modificación

1. Navegar hasta la ruta indicada anteriormente.
2. Abrir el archivo PubSubClient.h con un editor de texto (por ejemplo, Notepad++, VS Code, o el Bloc de notas).

Buscar la siguiente línea: `#define MQTT_MAX_PACKET_SIZE 128`

3. Reemplazarla por: `#define MQTT_MAX_PACKET_SIZE 51200`
4. Esto establece el tamaño máximo del mensaje MQTT a **51,200 bytes**, suficiente para transmitir imágenes en resolución VGA codificadas en Base64.

### 4.3 Justificación del cambio

El valor predeterminado de `MQTT_MAX_PACKET_SIZE` (128 bytes) es demasiado bajo para enviar imágenes, incluso en baja resolución. La cadena resultante en Base64 puede ocupar varios kilobytes, dependiendo del tamaño y calidad de la imagen capturada por la

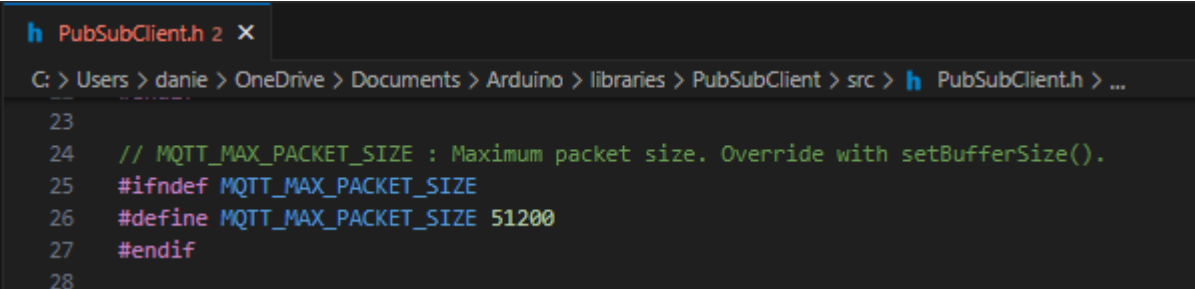
ESP32-CAM. Si no se ajusta este valor, el mensaje simplemente no se enviará, o el dispositivo puede reiniciarse por desbordamiento de memoria.

#### 4.4 Verificación posterior a la modificación

1. Guardar el archivo modificado.
2. Reiniciar el Arduino IDE para que los cambios surtan efecto.
3. Volver a compilar el código del proyecto de la ESP32-CAM. Si el cambio fue exitoso, el dispositivo podrá enviar imágenes grandes sin errores de transmisión.

#### Figura 5

*Modificación del tamaño máximo de paquete en PubSubClient.h para permitir envío de imágenes grandes (Arduino, s.f.).*



```
h PubSubClient.h 2 x
C:\Users\> danie > OneDrive > Documents > Arduino > libraries > PubSubClient > src > h PubSubClient.h > ...
23
24 // MQTT_MAX_PACKET_SIZE : Maximum packet size. Override with setBufferSize().
25 #ifndef MQTT_MAX_PACKET_SIZE
26 #define MQTT_MAX_PACKET_SIZE 51200
27 #endif
28
```

## 5. Conexión física del circuito

La correcta conexión de los sensores y actuadores es fundamental para el funcionamiento estable del sistema. A continuación se describen los esquemas de conexión de cada microcontrolador: el Arduino XIAO ESP32-C3 y la ESP32-CAM AI Thinker, incluyendo referencias a los pines GPIO y consideraciones eléctricas.

### 5.1 Conexiones del Arduino XIAO ESP32-C3

Este microcontrolador se encargará de la lectura ambiental mediante el sensor DHT22, y del control de dos dispositivos: un ventilador y una bombilla, activados a través de salidas digitales.

Distribución de pines:

Componente	Conexión	Pin en el XIAO ESP32-C3
Sensor DHT22	Señal	GPIO 4
Relé (para bombilla)	Control	GPIO 2
Ventilador	Control	GPIO 3
Alimentación común	VCC	3.3V o 5V del XIAO
Tierra común	GND	GND

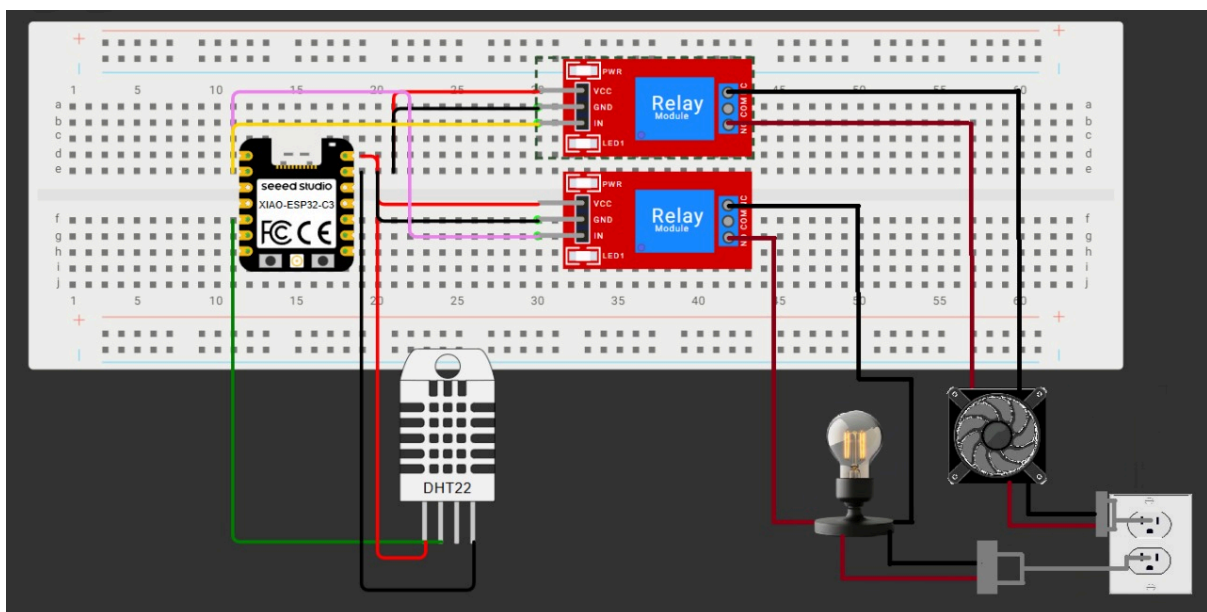
Importante: Asegúrate de verificar el tipo de relé utilizado. Si es de 5V, deberá conectarse a una fuente de alimentación externa o usar el pin de 5V del XIAO si está alimentado vía USB. El XIAO puede entregar una corriente limitada.

Consideraciones:

- El sensor DHT22 puede alimentarse con 3.3V o 5V, pero se recomienda usar 3.3V si se desea proteger el pin de entrada del XIAO.
- Se recomienda colocar una resistencia pull-up de 10k $\Omega$  entre VCC y la señal del DHT22 para mejorar la estabilidad de lectura.
- Si el ventilador o la bombilla requieren más corriente de la que puede entregar el microcontrolador, deben manejarse mediante relés o transistores MOSFET.

**Figura 6**

*Esquema de conexiones físicas del Arduino XIAO ESP32-C3.*



## 5.2 Conexiones de la ESP32-CAM (AI Thinker)

La ESP32-CAM se encargará exclusivamente de capturar imágenes y transmitir las por MQTT. Este módulo requiere una conexión cuidadosa, ya que su alimentación es crítica y su consumo es más elevado que otros ESP32.

Alimentación:

Fuente de energía	Recomendación
USB (a través de adaptador FTDI)	Conexión típica para programación y pruebas
Fuente externa 5V	Recomendado para funcionamiento estable en campo

- **GND** debe estar conectada a tierra común.
- **5V** puede provenir de una fuente externa o del pin VCC del adaptador USB-TTL si soporta suficiente corriente (>500 mA).

Pines usados por la cámara (configurados en el código):

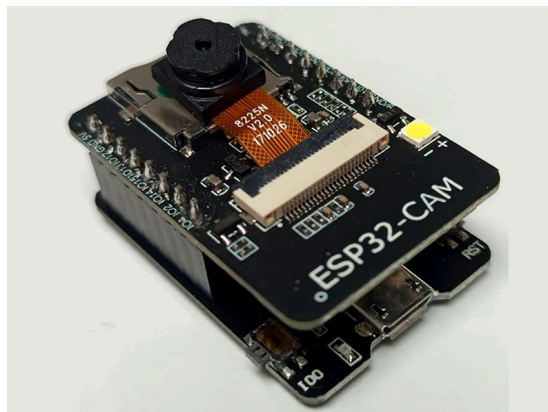
Señal de cámara	Pin asignado ESP32-CAM
D0 - D7	GPIOs 5, 18, 19, 21, 36, 39, 34, 35
XCLK	GPIO 0
PCLK	GPIO 22
VSYNC	GPIO 25
HREF	GPIO 23
SDA (SCCB)	GPIO 26
SCL (SCCB)	GPIO 27
PWDN	GPIO 32
RESET	-1 (no se usa)

Antena WiFi externa (opcional pero recomendada):

- Algunas versiones del ESP32-CAM tienen un conector **IPEX** para antena externa.
- Se debe desoldar el puente de antena para cambiar entre la antena PCB y la antena externa.
- Esto mejora significativamente la estabilidad de conexión WiFi, especialmente en ambientes cerrados como un invernadero.

### Figura 7

*Conexiones recomendadas para la ESP32-CAM AI Thinker, incluyendo asignación de pines para la cámara.*





## 6. Configuración de red WiFi y dirección del broker MQTT

Para que los microcontroladores puedan conectarse correctamente a la red local y comunicarse con el broker MQTT, es necesario modificar manualmente las siguientes variables en el código de cada dispositivo:

- Nombre de red WiFi (SSID)
- Contraseña WiFi
- Dirección IP del broker MQTT

Este paso debe realizarse antes de subir el código al microcontrolador.

### 6.1 Parámetros en el código del Arduino XIAO ESP32-C3

En el archivo .ino correspondiente al Arduino XIAO ESP32-C3, ubicar al inicio del código las siguientes líneas:

```
const char* ssid = "OneScreen Display 3415";  
const char* password = "ia20232023";  
const char* mqtt_server = "192.168.45.64";  
const int mqtt_port = 1887;
```

Personalización:

- Reemplazar "OneScreen Display 3415" por el nombre de la red WiFi a la que se conectará el dispositivo.
- Reemplazar "ia20232023" por la contraseña de esa red.
- Reemplazar "192.168.45.64" por la IP local del servidor MQTT (Mosquitto). Debe estar en la misma red que el ESP32-C3.
- Dejar el puerto 1887 si tu broker lo usa; en caso contrario, cambiarlo a 1883 (por defecto en Mosquitto).

## 6.2 Parámetros en el código de la ESP32-CAM

En el sketch de la cámara, ubicar estas líneas:

```
const char* ssid = "OneScreen Display 3415";  
const char* password = "ia20232023";  
const char* mqtt_server = "192.168.45.64";  
const int mqtt_port = 1887;
```

Recomendaciones:

- Los valores deben coincidir exactamente con los de tu red WiFi y tu broker Mosquitto.
- La IP del broker puede obtenerse ejecutando ipconfig (Windows) o ifconfig (Linux/macOS) en la máquina donde se ejecuta Mosquitto.
- Si usas una IP dinámica, considera reservar una IP fija para el broker en la configuración del router.

## 6.3 Prueba de conectividad

Una vez cargado el código con los parámetros correctos:

- Verifica en el monitor serial que ambos dispositivos se conecten a WiFi.
- Si la conexión falla, el dispositivo mostrará puntos suspensivos (....) en bucle.
- Si se conecta exitosamente, verás:

```
✓ WiFi conectado  
IP local: 192.168.45.XX
```

## 7. Subida del código

La carga del código a los microcontroladores Arduino XIAO ESP32-C3 y ESP32-CAM AI Thinker se realiza desde el Arduino IDE, seleccionando correctamente la placa, el puerto y los parámetros de compilación. También se ofrecen recomendaciones para resolver errores comunes como timeouts o fallos de conexión.

### 7.1 Subida del código al Arduino XIAO ESP32-C3

Pasos para cargar el código:

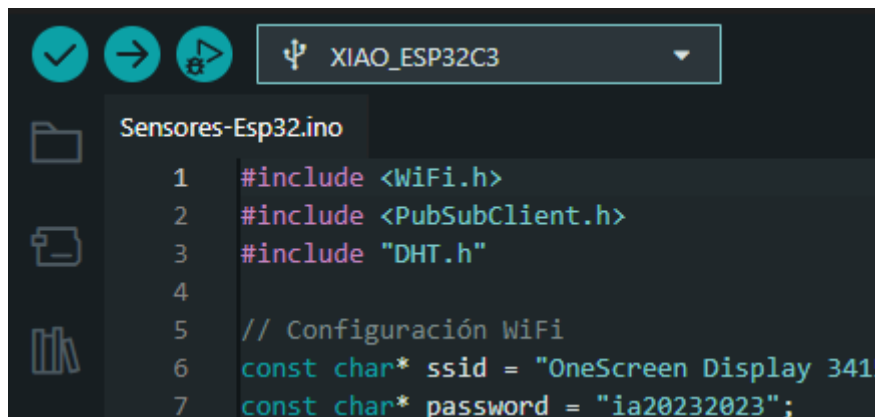
1. Conectar el XIAO ESP32-C3 por USB a la computadora.
2. En el Arduino IDE, ir a Herramientas > Placa y seleccionar:  
Sseed XIAO ESP32C3
3. Configurar los siguientes parámetros en el menú Herramientas:
  - Velocidad de carga: 115200
  - Partition Scheme: Default 4MB with spiffs (si está disponible)
  - Puerto: Seleccionar el que aparece como "USB to UART" o similar
4. Abrir el sketch del XIAO y hacer clic en el botón "Subir" (ícono de flecha).

Problemas comunes:

- Timeout al subir código: Si el dispositivo no responde al intentar cargar, mantener presionado el botón BOOT (si está disponible) durante el inicio de la carga.
- Si el puerto no aparece, cambiar de cable USB o puerto USB.

**Figura 8**

*Parámetros correctos para cargar el código al XIAO ESP32-C3 desde el Arduino IDE.*



## 7.2 Subida del código a la ESP32-CAM (AI Thinker)

Conecte el USB tipo C al ESP32-CAM y siga las instrucciones.

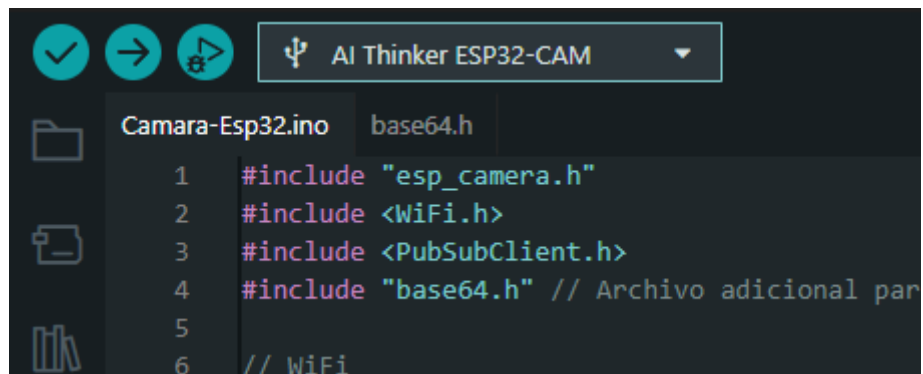
Pasos:

1. En el Arduino IDE, seleccionar AI Thinker ESP32-CAM como placa:
  - a. Herramientas > Placa > ESP32 Arduino > AI Thinker ESP32-CAM
2. Configurar:
  - a. Velocidad de carga: 115200
  - b. Partition Scheme: Default 4MB with spiffs
  - c. Puerto: El correspondiente al adaptador FTDI
3. Presionar Upload.
4. Durante el inicio de la carga, si da error:
  - a. Presionar el botón RESET en la ESP32-CAM (si tiene).
5. Una vez cargado, retirar el puente entre IO0 y GND y resetear para ejecutar el programa.

Problemas frecuentes:

**Figura 9**

*Parámetros correctos para cargar el código al ESP32-CAM desde el Arduino IDE.*



## 8. Verificación de funcionamiento

Una vez cargado el código en ambos microcontroladores, es fundamental verificar que:

- La conexión WiFi sea exitosa
- El broker MQTT esté accesible
- Los sensores y actuadores respondan
- Las imágenes se envíen correctamente


Esto se hace mediante el monitor serial del Arduino IDE, que permite visualizar mensajes de diagnóstico e interacción del sistema.

### 8.1 Verificación en el Arduino XIAO ESP32-C3

1. Conectar el XIAO al computador y abrir el **Monitor Serial** (Ctrl+Shift+M).
2. Configurar la velocidad en **115200 baudios**.

Salidas esperadas:

Confirmación de conexión WiFi:

 WiFi conectado


- Conexión al broker MQTT:

Conectando al broker MQTT...¡Conectado!


- Lecturas periódicas del sensor DHT22:

 Temp: 27.45 °C,  Humedad: 56.20 %

- Datos enviados en formato JSON:

 Datos enviados por MQTT

- Recepción de comandos MQTT:

 Mensaje recibido en [invernadero/bombilla]: auto  
 Modo automático activado

Estado de actuadores en modo automático:

Según el código, dependiendo de la temperatura:

- Si  $>30^{\circ}\text{C}$ : se activa el ventilador.
- Si  $<20^{\circ}\text{C}$ : se enciende la bombilla.
- En otros casos: ambos apagados.

## 8.2 Verificación en la ESP32-CAM AI Thinker

1. Abrir el monitor serial tras cargar el código.
2. Configurar también a 115200 baudios.

Salidas esperadas:

Conexión WiFi:

Conectando a WiFi...  
 WiFi conectado  
 IP local: 192.168.45.XX


- Conexión MQTT:

Conectando al broker MQTT...conectado

- Captura y envío de imagen cada 10 segundos:

 Imagen enviada por MQTT

- Si falla, se verá:

 Fallo al enviar imagen (reintento 1)

El sistema reintentará el envío automáticamente hasta lograrlo.

### 8.3 Validación desde el broker

Para verificar que los datos están llegando correctamente:

- Asegúrate de que el broker Mosquitto está ejecutándose.
- Usar una herramienta como MQTT Explorer para observar los mensajes en los siguientes tópicos:

Tópico MQTT	Contenido esperado
invernadero/sensores	JSON con temperatura y humedad
invernadero/bombilla	Mensajes de control (auto/on/off)
invernadero/status	Estado en línea del ESP32-C3
invernadero/imagen	JSON con imagen codificada en Base64

## 9. Conclusiones y recomendaciones finales

El sistema desarrollado permite implementar una solución IoT funcional para el monitoreo ambiental y visual de un invernadero hidropónico, utilizando tecnología de bajo costo y alta eficiencia como el Arduino XIAO ESP32-C3 y la ESP32-CAM AI Thinker, comunicados por MQTT a un broker local. A lo largo de este manual se detallaron todos los pasos necesarios para su implementación, desde la instalación del entorno hasta la verificación en tiempo real.

### 9.1 Conclusiones clave

- El uso de MQTT permite una comunicación ligera, estable y escalable, ideal para sistemas distribuidos con múltiples dispositivos.
- La división de tareas entre sensores (ESP32-C3) y cámara (ESP32-CAM) optimiza recursos y simplifica el diseño modular del sistema.



- La codificación en Base64 fue una estrategia adecuada para transmitir imágenes vía MQTT, a pesar de sus limitaciones de tamaño.
- Las condiciones de control automatizado del ventilador y la bombilla son fáciles de ajustar y escalar, permitiendo adaptar el sistema a distintos tipos de cultivos o ambientes.

## 9.2 Recomendaciones técnicas

### a) Asegurar la disponibilidad del broker MQTT (Mosquitto)

- El broker debe estar activo en la red local antes de iniciar los dispositivos.
- Puede ejecutarse en una computadora, Raspberry Pi o servidor dedicado.
- Se recomienda configurar el broker con una IP fija o hostname reconocible.

### b) Visualización con herramientas de monitoreo

- Se recomienda utilizar MQTT Explorer o Node-RED para observar en tiempo real los mensajes publicados y sus contenidos.
- Estas herramientas permiten comprobar el tráfico en los tópicos MQTT, facilitando el debugging y la supervisión.

### c) Evitar errores comunes

- Errores de conexión: Verificar SSID, contraseña y alcance de la red WiFi.
- Errores de carga: Revisar conexión de pines, drivers USB, cables y configuración del puerto.
- Errores de memoria: Usar imágenes de baja resolución (como VGA) y ajustar la calidad JPEG para reducir el tamaño.
- Reinicios aleatorios en la ESP32-CAM: Usar fuentes de alimentación confiables de 5V y mínimo 500 mA.

### 9.3 Escalabilidad y mejoras futuras

- Agregar sensores adicionales: luminosidad, pH, EC (electroconductividad), CO<sub>2</sub>.
- Incorporar control remoto mediante dashboard web o app móvil conectada a los tópicos MQTT.
- Usar microcontroladores con más recursos si se desea transmitir imágenes más grandes o con mayor frecuencia.
- Implementar almacenamiento local o en la nube para históricos de datos e imágenes.

## Referencias

- Arduino. (s.f.). *Arduino IDE software*. Arduino. Recuperado el 28 de mayo de 2025, de <https://www.arduino.cc/en/software>
- Espressif Systems. (s.f.). *ESP32 Arduino core documentation*. GitHub. Recuperado el 28 de mayo de 2025, de <https://github.com/espressif/arduino-esp32>
- Seeed Studio. (s.f.). *XIAO ESP32C3 product page*. Seeed Studio Wiki. Recuperado el 28 de mayo de 2025, de [https://wiki.seeedstudio.com/XIAO\\_ESP32C3\\_Getting\\_Started/](https://wiki.seeedstudio.com/XIAO_ESP32C3_Getting_Started/)
- Mosquitto. (s.f.). *Eclipse Mosquitto MQTT Broker*. Eclipse Foundation. Recuperado el 28 de mayo de 2025, de <https://mosquitto.org/>
- Adafruit. (s.f.). *DHT sensor library for Arduino*. GitHub. Recuperado el 28 de mayo de 2025, de <https://github.com/adafruit/DHT-sensor-library>
- Knolleary, N. (s.f.). *PubSubClient MQTT library for Arduino*. GitHub. Recuperado el 28 de mayo de 2025, de <https://github.com/knolleary/pubsubclient>
- MQTT Explorer. (s.f.). *MQTT Explorer - GUI Client for MQTT*. MQTT Explorer. Recuperado el 28 de mayo de 2025, de <https://mqtt-explorer.com/>
- Random Nerd Tutorials. (s.f.). *ESP32-CAM Video Streaming and Face Recognition with Arduino IDE*. Recuperado el 28 de mayo de 2025, de <https://randomnerdtutorials.com/esp32-cam-video-streaming-face-recognition-arduino-ide/>