

Memoria Proyecto Final



Grado en Ingeniería Robótica



Universitat d'Alacant
Universidad de Alicante

Sergio González Rodríguez
Laura Bermejo Mateos
Pablo Gorbarán, Jr

Comunicaciones
30/05/2024

Introducción

En la era digital actual, la seguridad y la vigilancia se han convertido en aspectos cruciales tanto en entornos residenciales como comerciales. La integración de tecnologías avanzadas como la visión por computadora y el Internet de las Cosas (IoT) ha permitido desarrollar sistemas de vigilancia más eficientes y automatizados.

El sistema propuesto se compone de tres cámaras ESP32, dispositivos que combinan capacidades de procesamiento y conectividad, ideales para aplicaciones de IoT. Estas cámaras están configuradas para transmitir video en tiempo real, el cual es monitoreado manualmente a través de Home Assistant, una plataforma de automatización del hogar que permite integrar y gestionar múltiples dispositivos inteligentes.

Además del monitoreo manual, el proyecto incorpora un componente de visión artificial mediante un script en Python. Este script utiliza técnicas avanzadas de procesamiento de imágenes y aprendizaje automático para detectar la presencia de personas en las secuencias de vídeo capturadas por las cámaras ESP32. La detección automatizada de personas es una funcionalidad clave que mejora significativamente la eficacia y la capacidad de respuesta del sistema de vigilancia.

Materiales

Placa y cámara

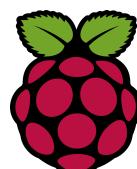
Se ha optado por trabajar con una placa de la familia ESP32 dada su flexibilidad y accesibilidad en el mercado. En concreto se ha adaptado el sistema de comunicaciones al conjunto [ESP32-CAM-MB](#) que consta de la propia placa ESP32, una base de conexión (que facilita la subida del código y la alimentación) y una cámara del modelo OV2640.



Este conjunto de cámara y placa es mucho más barato que otras opciones de mercado por lo que se adapta mejor a las condiciones del proyecto propuesto. Comparándolas con el resto del mercado encontramos que este set cuesta prácticamente la mitad que las cámaras con detección de personas de las marcas más asequibles como pueden ser [Imou](#) o [Tapo](#).

Raspberry

Para el control central del sistema de vigilancia se plantea el uso de una Raspberry Pi. El modelo depende de los objetivos finales del proyecto y la potencia de la inteligencia artificial implementada para la detección, pero se considera que con una Raspberry Pi 3 (si la IA es simple y está optimizada) se puede llevar adelante.



La Raspberry será el centro del sistema de comunicaciones y se encargará del procesado de las imágenes y la gestión de las comunicaciones entre la información obtenida de las cámaras y las plataformas que se consideren (ya sea Telegram, Home Assistant o un propio display por monitor directamente desde la Raspberry).

Router

Para llevar a cabo la conexión IP de los distintos dispositivos empleados en el proyecto es necesario disponer del servicio de un router. Para la demostración presencial se acondicionó un portátil para que cumpliera la función del router a falta de uno. Las únicas condiciones que necesita cumplir nuestro router es que trabaje con una frecuencia de 2.4GHz como puede ser el [TP-LINK TL-WR820N](#). En el caso de la demostración presencial se fuerza a la frecuencia a cumplir con esta restricción.

Propiedades		
Propiedades de red		
Nombre:	DESKTOP-3FNHAPR 3439	Editar
Contraseña:	3w281X9/	
Banda:	2.4 GHz	
Dispositivos conectados:	5 de 8	
Nombre del dispositivo	Dirección IP	Dirección física (MAC)
Laura	192.168.137.184	ec:63:d7:cd:40:fb
esp32-D381EC	192.168.137.144	08:f9:e0:d3:81:ec
M2101K7BNY	192.168.137.3	9e:17:9d:b0:98:65
esp32-88D5B0	192.168.137.30	80:7d:3a:88:d5:b0
esp32-EBCA2C	192.168.137.170	08:f9:e0:eb:ca:2c

Podemos comprobar que se puede acceder a la información de las IP conectadas al router y que se puede configurar la frecuencia deseada sin problemas.

Desarrollo

Conexión de las cámaras: Asignación IP

Se ha configurado en el IDE de arduino, donde se ha instalado la configuración de la placa ESP-32.

Se ha partido del ejemplo de la librería de la cámara, hemos tenido problemas a la hora de conectarnos correctamente, ya que estábamos usando una red WiFi 5G, y no conseguimos que la cámara se conectaría correctamente, en el monitor Serial de Arduino nos salen caracteres raros e inteligibles.

Hemos usado un ordenador como router, ya que de esa manera no hay restricciones de uso de red como con routers comerciales.

Tras varias pruebas hemos cambiado a una red 2.4G y funciona bien, obtenemos la siguiente información.

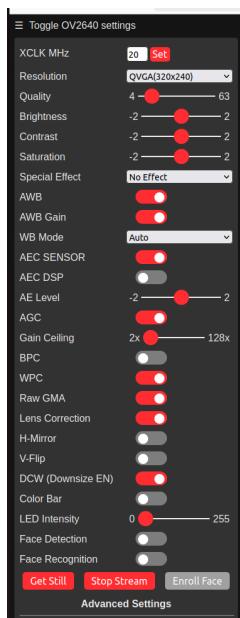
```

COM3
[REDACTED] Enviar
ets Jul 29 2019 12:21:46
rst:0x1 (POWERON_RESET),boot:0x3 (DOWNLOAD_BOOT(UART0/UART1/SDIO_REI_RXO_V2))
waiting for download
ets Jul 29 2019 12:21:46
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1344
load:0x40078000,len:13964
load:0x40080400,len:3600
entry 0x400805f0

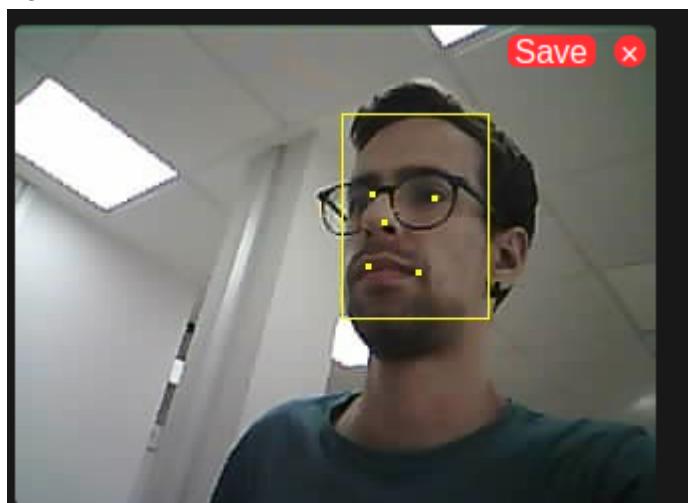
.
WiFi connected
Camera Ready! Use 'http://192.168.137.231' to connect

```

A través de la IP suministrada podemos acceder a la información y utilidades de la cámara.



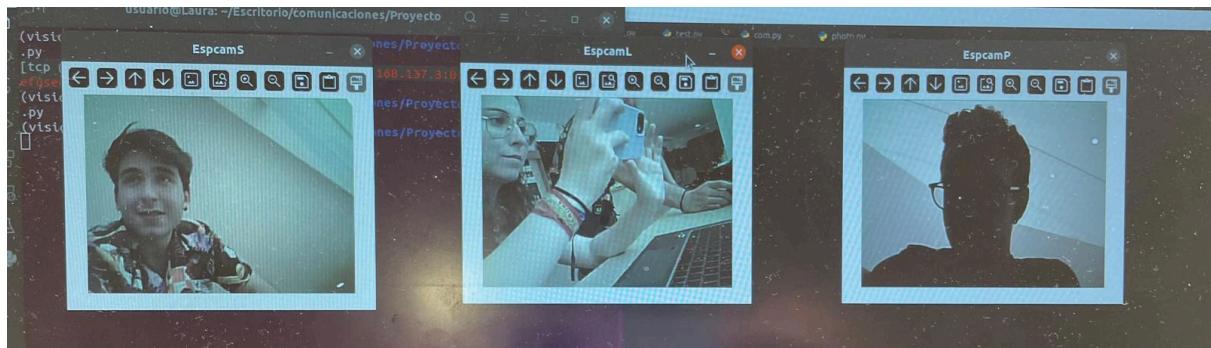
Se ha probado la detección de caras propia de la cámara, pero para un sistema de vigilancia como el nuestro necesitamos detectar no caras, si no personas.



Código Arduino

Streaming Python mediante dirección IP

Para la realización de un streaming mediante python se utilizaron las direcciones IP de las cámaras obtenidas previamente. Es importante destacar que para obtener el streaming deseado el enlace de lectura de las IP debe tener el siguiente formato "http://IP_CAMARA:81/stream". Una vez declarado el enlace de acceso al stream se lee mediante el uso de la librería OpenCV dado que conocemos que se trata de una imagen. Posteriormente en el código python se puede procesar el streaming frame a frame y tratar las imágenes de la forma deseada. En la siguiente imagen se puede observar como se mostraría un streaming por pantalla.



En este proyecto hemos optado por realizar una detección de personas en cada frame obtenido del streaming utilizando un modelo de inteligencia artificial entrenado basado en YOLO4. Al hacerlo un programa de python nos permite gestionar al mismo tiempo tantas cámaras como queramos y nos brinda flexibilidad en la vigilancia al poder modificar el modelo de detección de forma que permita adaptar la detección de la cámara a las necesidades del usuario: Esta flexibilidad no sería posible con otros tipos de dispositivos que además de tener un mayor coste, solo permiten detección de personas y reconocimiento facial. En la siguiente imagen se observa el streaming junto con la detección de personas.



Conexión Python con Telegram

Además de lo comentado en el apartado anterior la conexión de las cámaras con python nos permite gestionar lo que queremos hacer con las imágenes obtenidas, en este caso se ha optado por la opción de mandar las imágenes deseadas a un bot de Telegram junto con un identificador de la cámara a la que pertenecen. En este caso se mandaría una imagen cuando el programa detectase una persona pero debido a la alta carga computacional del modelo de IA utilizado se optó por hacer una demo de cada una de las partes por separado.

Para la conexión con Telegram se realiza una publicación en un bot de Telegram del cual conocemos el token y el chat ID. Python nos permite realizar esta conexión con el uso de la librería requests. Cabe destacar que las imágenes enviadas por telegram son imágenes previamente descargadas del streaming al ordenador debido a que la librería utilizada para el procesamiento de las imágenes funciona mediante arrays que no podrían ser enviados por telegram. Las imágenes previamente descargadas se cargan en el programa como archivo y se envían haciendo uso de la URL de la API de telegram con un post de la librería requests mencionada anteriormente. En la siguiente imagen se observan las imágenes y textos recibidos por telegram.

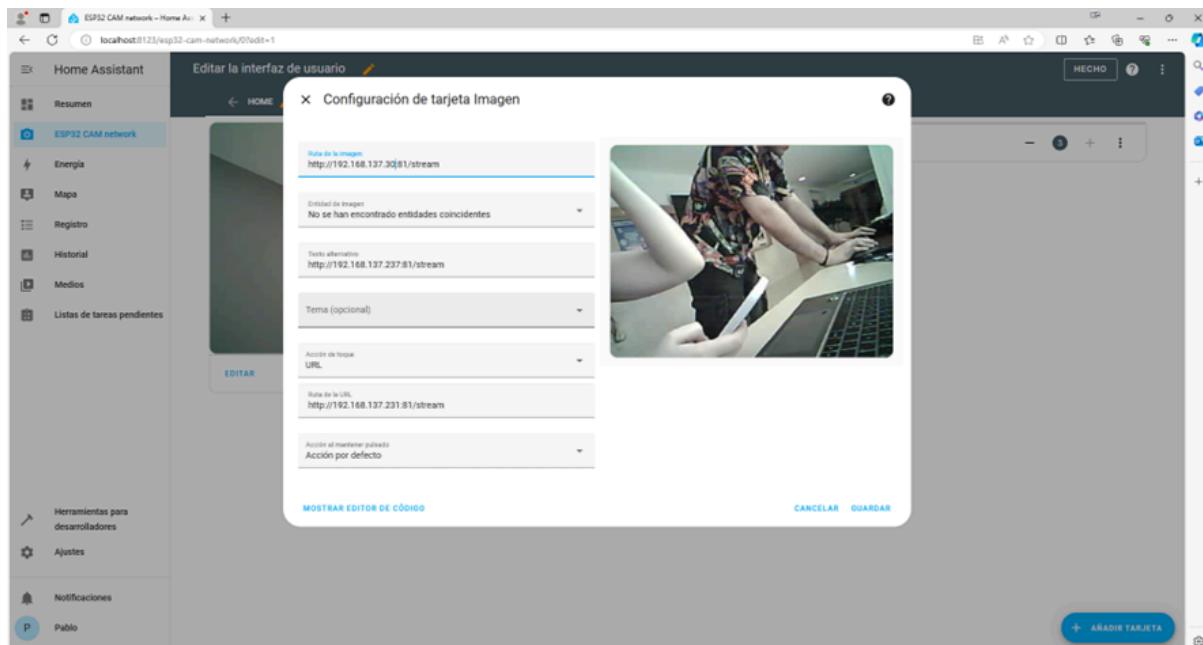


Home Assistant: Monitoreo Manual

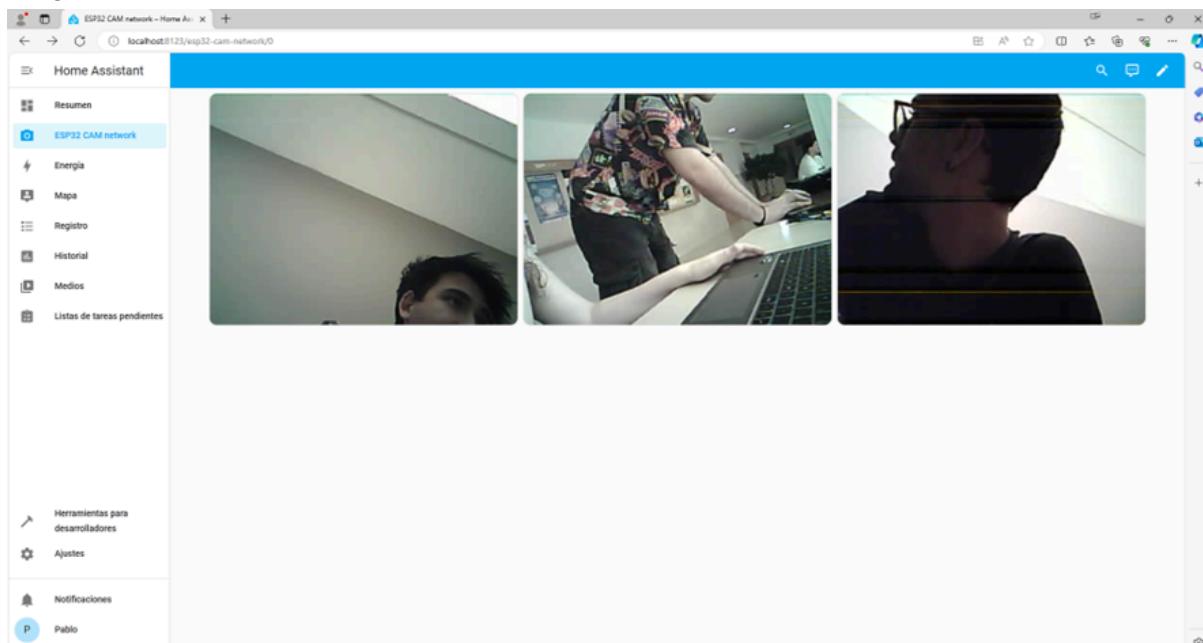
Se han agregado las direcciones en Home Assistant para poder realizar un monitoreo manual, al que puedas acceder desde el móvil.

Video en tiempo real en Home Assistant

La visualización del streaming de las cámaras en Home Assistant es muy sencilla, se empieza por crear un dashboard en el que se podrán emitir los videos de las distintas cámaras. Dentro de este dashboard se crea una tarjeta imagen a la que se le referencia la imagen a mostrar con el link al stream de la cámara deseada.



Repitiendo este proceso para todas las cámaras obtenemos un dashboard con las imágenes en tiempo real de estas.



También está la posibilidad de realizar el monitoreo mediante la aplicación para smartphones



Protocolo MQTT

También se ha pensado en otro enfoque dentro del mismo proyecto, este centra el estudio de las imágenes de cada cámara en sus respectivas placas y no en la Raspberry. Con esto las cámaras se centrarían en la detección o reconocimiento de personas y la Raspberry se enfocaría exclusivamente en la gestión de comunicaciones y selección de acciones.

Para poder llevar esto a cabo se pensó en implementar un modelo de comunicaciones centrado en el protocolo MQTT. El router sería el broker de la comunicación entre las distintas cámaras y la Raspberry y existirían diversos topics que ayudarían a la gestión del paso de datos entre estas.

Las ventajas que nos aporta la aplicación del protocolo MQTT en este enfoque es que se puede crear un topic por cada una de las cámaras implementadas en el sistema y que cuente con los subtopics que consideremos necesarios (detección_de_persona, reconocimiento_sujeto_a, reconocimiento_sujeto_b...). Del mismo modo, la Raspberry contaría con sus propios topics y podría solicitar a las distintas cámaras que tomen acciones como sacar una captura del frame en el que se encuentran o cambiar entre modos de detección y reconocimiento.

Estos flags ayudarían a la raspberry a poder, por ejemplo, tomar una imagen de la cámara deseada cuando esta detecte a una persona y mandarla junto con un mensaje identificativo por telegram al usuario (función implementada anteriormente).

Gracias a la reducción de la carga de trabajo en la Raspberry conseguida gracias a este enfoque se puede optar por versiones de Raspberry más asequibles como puede ser la Zero.

Cabe tener en cuenta que pese a obtener una carga de trabajo reducida en la Raspberry este modelo se apoya del software ya implementado en las cámaras con el que se pierde flexibilidad ya que solo se podrían detectar y reconocer rostros.

Conclusiones

A lo largo del proyecto se han probado distintos protocolos de comunicación con un material muy económico y accesible.

No solo se ha diseñado una red de cámaras conectadas a una Raspberry que gestiona el sistema, sino que se han propuesto distintas funciones de vigilancia (ya sea reconocimiento o detección) y de display de los datos (imágenes por Telegram, dashboard de Home Assistant tanto en la app de móvil como en la web, acceso a las cámaras a través de IP).

Es un proyecto con una gran flexibilidad y muy actual que nos ha permitido, gracias al material empleado, no solo diseñar el modelo de comunicaciones, sino que llevarlo a cabo nosotros mismos permitiéndonos encontrar trabas y detalles en el proceso de comunicación que de otro modo no hubieran sido posibles.

Esta propuesta da pie también a varias ampliaciones como el acceso a las imágenes desde un entorno no perteneciente a la IP del router, cambiar los paradigmas de detección de la IA implementada en la prueba para poder detectar eventos que no sean únicamente personas o añadir otra clase de sensores y actuadores controlables por IP.

Bibliografía

Video explicativo del uso de la ESP-32 CAM

<https://www.youtube.com/watch?v=LOqVle9cnW8>

Programa ESP32

https://programarfacil.com/esp32/esp32-cam/#Como_programar_tu_ESP32_CAM

Programa python:

<https://lascosasdedani.com/como-conectarnos-a-un-modulo-esp32-cam-desde-python>

Link de compra de la cámara:

MQTT:

<https://www.prometec.net/instalar-mosquitto-windows/>