|  |  |
| --- | --- |
| Monitoreo en tiempo real con ESP32-CAM  Sistema de vigilancia y almacenamiento de imágenes | Carlos Javier Roviralta Rodríguez  Ronny Toledo Ramos  Andrés Piña Cardoso  Ingeniería Automática |

**Introducción**

En la actualidad, la seguridad en hogares e instituciones es una prioridad debido a la frecuencia de accidentes laborales y robos. Los métodos tradicionales de seguridad, como candados y cerrojos, han sido reemplazados por tecnologías avanzadas que incluyen cámaras, sensores y alarmas, mejorando significativamente la protección de las propiedades.

El avance en la automatización ha permitido el desarrollo de sistemas de vigilancia más eficientes y económicos. En los últimos años, el uso de hardware libre ha ganado popularidad en aplicaciones relacionadas con el Internet de las Cosas (IoT) [1]. Estos dispositivos ofrecen ventajas como bajo costo, gran cantidad de información disponible, entorno amigable para principiantes y versatilidad en su uso. Los sistemas de vigilancia no son una excepción, y el uso de hardware libre resulta mucho más económico y menos complejo que los métodos tradicionales.

El ESP32-CAM es uno de los dispositivos más utilizados para estos fines. Es una pequeña y económica tarjeta que incorpora el módulo ESP32-S, capacidades Wi-Fi y un módulo de cámara, lo que la hace ideal para sistemas de captura de imágenes en tiempo real. Es posible utilizar microprocesadores de hardware libre, como el ESP32-CAM, junto con bases de datos como Google, para recibir imágenes en tiempo real [2]. También se puede añadir una cámara USB a un Raspberry Pi o usar una cámara bajo el control de un Arduino UNO y un sensor infrarrojo pasivo (PIR).

Basándonos en estas aplicaciones, se plantea la siguiente hipótesis: con el uso de módulos ESP32-CAM y sensores PIR para la detección de presencia, es posible diseñar un sistema de vigilancia eficiente y económico que cumpla con los requerimientos necesarios. Además, este sistema ofrece varias ventajas, desde la facilidad de instalación hasta una configuración flexible y un bajo requerimiento de almacenamiento.

El objetivo general del trabajo es diseñar un sistema de vigilancia y almacenamiento de imágenes recopiladas en un horario determinado, aprovechando las utilidades que brinda el ESP32-CAM para la captura y almacenamiento de imágenes.

**Objetivos Específicos:**

1. Investigar las mejores opciones de hardware libre y las ventajas que estos ofrecen, especialmente el ESP32-CAM.
2. Programar el sistema de vigilancia utilizando software libre de Arduino.
3. Montar el dispositivo en una habitación para probarlo posteriormente.
4. Realizar pruebas al sistema de vigilancia para comprobar su correcto funcionamiento.
5. Analizar los resultados obtenidos de las pruebas realizadas.

**Desarrollo**

1. **Marcó teórico relevante:**

ESP32-CAM (Ilustración 1):

Es un módulo que combina las capacidades de un microcontrolador ESP32 con una cámara de video. Está diseñado para proyectos donde se requiere captura de imágenes o video, así como conectividad WiFi y Bluetooth, ideal para aplicaciones de Internet de las Cosas (IoT).

PIRHC-SR501 **(**Passive Infrared) (Ilustración 2):

Es un dispositivo electrónico que detecta movimiento. "PIR" significa infrarrojo pasivo, lo que indica que el sensor detecta las variaciones en la radiación infrarroja (calor) que emiten los objetos en su campo de visión.

Arduino:

Es una plataforma de electrónica de código abierto (open-source) basada en hardware y software fáciles de usar. Está diseñada para que tanto aficionados como profesionales puedan crear proyectos interactivos y objetos autónomos. Aquí algunos puntos clave:

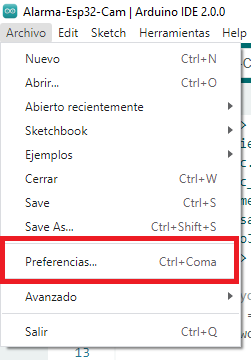
Componentes Principales:

* Placa Arduino (Ilustración 3): Es el hardware que contiene un microcontrolador programable. Hay varios modelos de placas Arduino como el Uno, Mega, Nano, etc., cada uno con diferentes capacidades y características.
* IDE de Arduino (Ilustración 4): Es el entorno de desarrollo integrado, un software gratuito donde puedes escribir, compilar y cargar el código a la placa Arduino.
* Lenguaje de Programación: Se basa en C/C++, con muchas bibliotecas y funciones predefinidas para facilitar el desarrollo de proyectos.

1. **MATERIALES Y MÉTODOS**

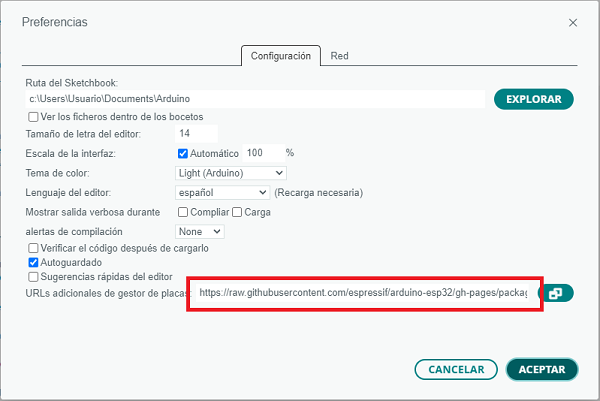
* **Componentes**
* Módulo Esp32-Cam para la toma y el almacenamiento de imágenes.
* Sensor de Movimientos PIR HC-SR501 para la detección de movimiento.
* Arduino Uno (Software Arduino IDE)
* **Agregar tarjeta Esp32**

Para agregar la url de la tarjeta del Esp32 debemos ir al Arduino IDE y seleccionar Archivo -> Preferencias

[](file:///C:\Users\Carlos%20Javier\Desktop\Proyecto%20IA\alarm-esp32cam_README.md%20at%20main%20·%20electrodeuna_alarm-esp32cam%20·%20GitHub_files\198603503-9bda20db-db92-4efb-98f4-a3a74ac37f02.png)

Luego en "Gestor de URLs Adicionales de Tarjeta" agregamos:

[*https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package\_esp32\_index.json*](https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json)

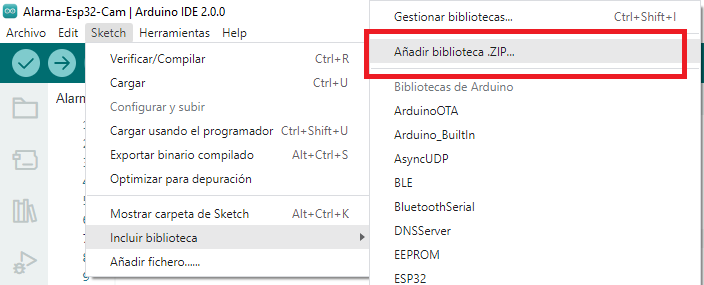
[](file:///C:\Users\Carlos%20Javier\Desktop\Proyecto%20IA\alarm-esp32cam_README.md%20at%20main%20·%20electrodeuna_alarm-esp32cam%20·%20GitHub_files\198605632-ed09f245-3cc8-48da-a1cb-00f2c3a3d3b5.png)

Si ya tenemos una url agregada previamente, ponemos la nueva url con una coma ","

* **Agregar librerías**

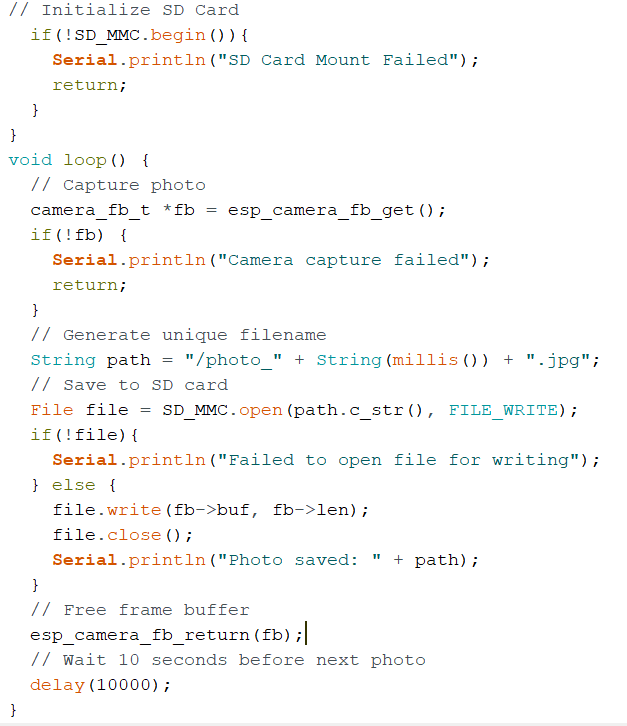
Las librerías utilizadas para este proyecto son "ArduinoJson", "UniversalTelegramBot" y “Esp32”, se instalan desde el gestor de librería del Arduino IDE. También se pueden descargar directamente e instalarlas en Arduino IDE desde Programa (o Sketch) -> Incluir Librería -> Añadir biblioteca .ZIP

UniversalTelegramBot: [*https://github.com/witnessmenow/Universal-Arduino-Telegram-Bot*](https://github.com/witnessmenow/Universal-Arduino-Telegram-Bot)

[](file:///C:\Users\Carlos%20Javier\Desktop\Proyecto%20IA\alarm-esp32cam_README.md%20at%20main%20·%20electrodeuna_alarm-esp32cam%20·%20GitHub_files\198614626-a11b4904-d129-4d98-a33d-fcf4c0be3fed.png)

* **Para ver las imágenes tomadas por el ESP32-CAM se cuenta con dos métodos:**

1. Almacenamiento en Tarjeta SD



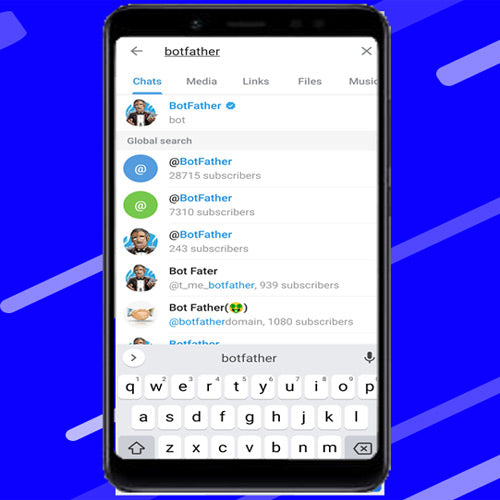
Código para inicializar la tarjeta SD y guardar las imágenes en ella.

1. Almacenamiento en Telegram (Bot)

* Creando el Bot de Telegram

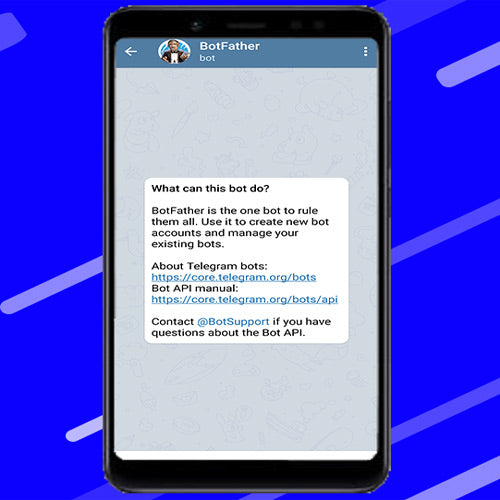
>Como primer paso se instaló la App y se creamos una cuenta.

> Luego se buscó "BotFather" y se hizo clic en el chat de BotFather, tal como se mostró. También se pudo abrir el enlace t.me/botfather [2] desde el celular.Botfather es un Bot preconstruido de Telegram que te permite administrar tus propios bots.



Vía: *IoTDesignPro*

> Se hizo clic en empezar y se seleccionó /newbot.



Vía: IoTDesignPro

> Se le dio un nombre al Bot en "name" y se creó un nombre de usuario en "username". Si se logró crear el bot correctamente, se recibió un mensaje con el enlace hacia el nuevo bot y un bot token. Este token fue una identificación única del bot que permitió establecer la comunicación posteriormente.



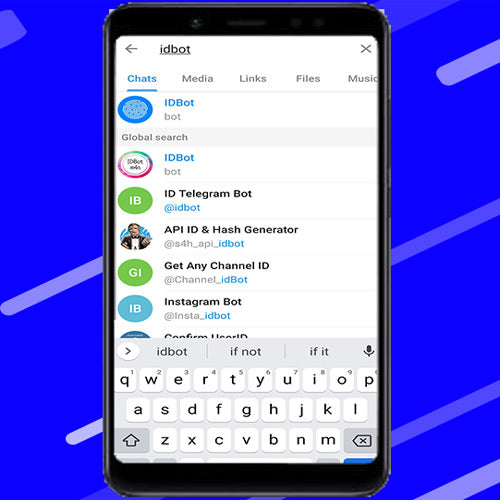
*Vía: IoTDesignPro*

* Configurando el id de Telegram

El ID de usuario de Telegram fue un número único para cada chat, grupo y usuario que le permitió a Telegram identificar usuarios y chats. En este proyecto, cualquiera que tenía el enlace al bot podía interactuar con él. Para prevenir accesos no autorizados, se encriptó el bot usando dicho ID. Haciendo esto, cada vez que la tarjeta ESP recibía un mensaje del bot, se revisaba primeramente si era nuestro usuario quien mandaba el mensaje, y solo se ejecutaba el comando cuando éramos nosotros.

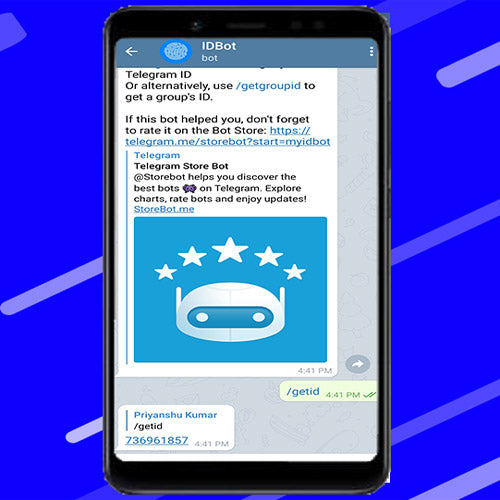
* Pasos para configurar el ID:

> Desde la cuenta, se buscó IDBot, o se abrió el enlace t.me/myidbot. [2]



*Vía: IoTDesignPro*

> Se envió un mensaje al bot con el comando /getid, lo cual dio el ID de usuario. Se anotó el ID, el cual se usaría después.



Vía: IoTDesignPro

* Instalando las librerías para usar el Bot
* Librería de Telegram

Para establecer la comunicación con el Bot, se usó la librería Universal de Telegram creada por Brian Logh, que proveyó una interfaz fácil de usar para la API de Telegram. Para instalarla, se necesitó descargarla primero desde aquí y luego, en el IDE de Arduino, se seleccionó Sketch > Include Library > Add.ZIP Library... Y listo. Para más detalles de la librería, se podía revisar la documentación en GitHub.

* Librería de ArduinoJson

Para instalar la librería de Json, se necesitó buscarla desde el administrador de librerías de Arduino. Se abrió Sketch > Include Library > Manage Libraries y se buscó ArduinoJson.

* **Programando la tarjeta ESP32**

Se tuvo que programar la tarjeta de modo que recibiera cualquier mensaje enviado por el bot, comparara el ID de usuario y encendiera o apagara el LED de acuerdo al mensaje recibido.

Para empezar, agreguemos las librerías instaladas previamente y luego se terminó de programar el código:

Incluimos todas las librerías necesarias, colocamos nuestro usuario y contraseña y luego introducimos el ID y Token obtenido anteriormente a través de IDBot y BotFather (Ilustración 5). Después se definieron las variables a emplear y se procedió comenzar la configuración del sensor PIR para la detección de movimiento(Ilustración 6).Luego se asignaron las variables, los textos a mostrar y los pines correspondientes(Ilustración 7 y 8).A continuación se establecieron los bucles para que la tarjeta supiera lo que debía hacer en caso de recibir la señal de movimiento detectado y las acciones que debía tomar para hacer llegar la imagen al bot de telegram configurado anteriormente(Ilustración 9 y 10).Finalmente se terminó de ultimar detalles sobre las prestaciones que brindara el bot (Ilustración 11 y 12).

1. **Conexiones**

El módulo ESP32-CAM se utiliza en el proyecto para capturar imágenes y almacenarlas en una tarjeta MicroSD. Este módulo incluye una cámara OV2640 de 2 MP y un LED integrado que funciona como flash durante la captura de imágenes. También cuenta con un puerto para conectar el dispositivo de almacenamiento mencionado, así como conexiones WiFi y Bluetooth. La placa se alimenta con 5V y tiene 8 pines de entrada y salida digitales generales, capaces de proporcionar un voltaje de 3.3V. Además, cuenta con una memoria RAM de 512 KB. Sus dimensiones físicas, 40 mm de largo por 27 mm de ancho, y su bajo consumo de energía lo hacen ideal para aplicaciones de IoT domésticas o de bajo costo.

El HC-SR501 es un sensor de movimiento del tipo PIR (Infrarrojo Pasivo) que consta de dos elementos principales. Por un lado, cuenta con un dispositivo emisor de señal diferencial que activa la señal de alarma en comparación con otros sensores. Este funcionamiento se logra mediante un circuito integrado BISS0001, que incluye amplificadores operacionales e interfaces electrónicas adicionales.

Además, el módulo permite realizar dos ajustes de sus funciones. El primero es la sensibilidad de la distancia de detección del PIR, ajustable mediante potenciómetros. El segundo es la capacidad de detección automática de luz, aunque esta función no está habilitada de fábrica.

La función de detección automática de luz se utiliza comúnmente en sistemas que encienden una luz cuando se detecta movimiento y la iluminación ambiental es baja, como durante la noche.

El HC-SR501 tiene un rango de detección de movimiento de 3 a 7 metros y un ángulo de apertura de hasta 90° y 110°. Esto permite una instalación flexible en diversos lugares, como paredes, techos o suelos, según las necesidades del proyecto.

El PIR posee un jumper mediante el cual se puede trabajar en 2 modos de

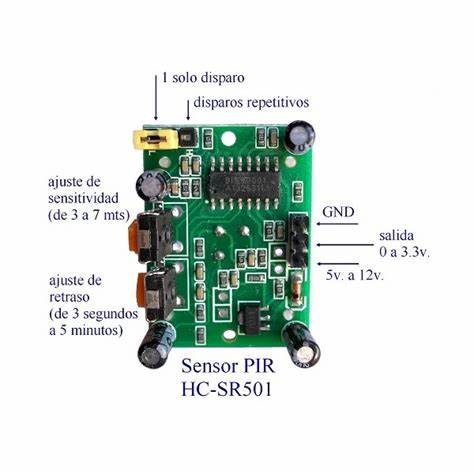
operaciones diferentes:

Modo Solo Disparo

En este modo, cuando el sensor detecta un movimiento, la salida se activa durante el periodo de tiempo ajustado (por ejemplo, 30 segundos). Durante estos 30 segundos, no se considerarán nuevos eventos.

Modo Disparos Repetidos

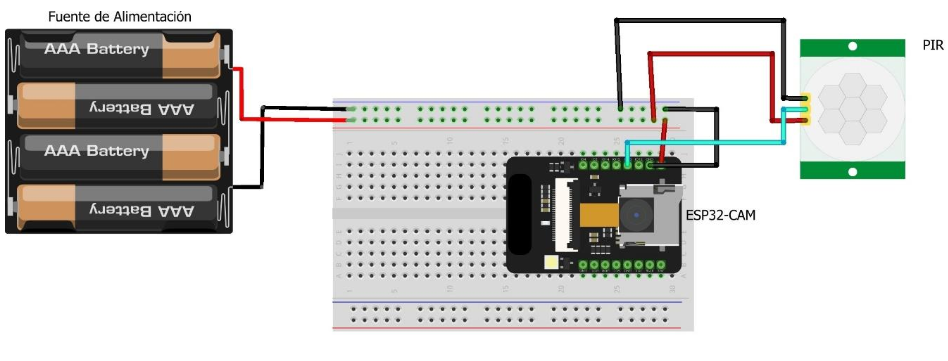
En este modo, cada evento genera un nuevo periodo de activación de 30 segundos. Si ocurre otro evento durante el intervalo de activación, se suman 30 segundos adicionales a la activación de la salida. Si la salida regresa a su estado inactivo, hay un lapso de 3 segundos antes de que el sensor pueda detectar nuevos eventos.



Para hacer funcionar correctamente el sistema, se deben seguir dos pasos. Primero, se debe programar el ESP32-CAM con el código necesario utilizando un Arduino UNO como puente entre el ESP32-CAM y la computadora. Esto es necesario porque el módulo ESP32-CAM no tiene un programador incorporado, por lo que el Arduino UNO actuará como adaptador UART-TTL. Para realizar esta tarea, se debe conectar el hardware como se muestra a continuación:

Para conectar el ESP32-CAM al Arduino UNO, se deben unir los pines U0R y U0T del ESP32-CAM con los pines Tx0 y Rx0 del Arduino UNO, respectivamente. Además, se deben conectar los pines GND y 5V de ambas placas. Finalmente, se deben hacer puentes entre los pines GND y Reset del Arduino UNO y el pin 0 y GND del ESP32-CAM (Ilustración 13). Una vez que el programa se haya cargado en el ESP32-CAM, se pueden deshacer todas estas conexiones y proceder al siguiente paso, donde se activa el programa.

Conexiones necesarias para el correcto funcionamiento del sistema de vigilancia diseñado utilizando la señal del sensor PIR para la detección de movimiento en la habitación.



Ambos dispositivos se alimentan con una fuente de 5V y el pin de salida del sensor se conecta al pin 13 del ESP32-CAM, por donde será recibida la señal cada vez que se detecte movimiento.

1. **Resultados y discusión:**
2. Resultados obtenidos:

Al finalizar el desarrollo del proyecto de monitoreo en tiempo real con ESP32-CAM se realizaron varias pruebas para comprobar el correcto funcionamiento del dispositivo, resultados que se muestran a continuación:

* Detección de movimiento mediante PIR:
* El PIR funciono de manera correcta, logrando detectar con gran rapidez los movimientos dentro de su rango y respondiendo de manera eficaz a ellos, enviando una señal al módulo de ESP32-CAM.
* Visualización de imágenes a través del ESP32-CAM:
* El módulo fue capaz de captar imágenes y capturarlas de manera eficiente.
* La entrega de imágenes a Telegram a través del módulo también funcionó correctamente y cabe destacar que con buena velocidad.

De forma general se recrearon diferentes situaciones para comprobar el funcionamiento de todas las partes en cooperativo obteniéndose buenos resultados ya que los módulos se comunicaron de manera satisfactoria entre ellos.

Situación 1

Aquí se realizó un movimiento del brazo donde solo la sombra entró en el campo del PIR, el cual captó con éxito el movimiento enviando la señal al módulo ESP32-CAM que se encargó de tomar la fotografía (Ilustración 9).

Situación 2

En esta ocasión se pasó el brazo por delante del PIR captando nuevamente el movimiento y enviando la señal para realizar la toma de la fotografíala (Ilustración 10).

Situación 3

Para esta situación se colocó una persona delante del sensor, él cual detecto con éxito el movimiento enviando con éxito la señal para que posteriormente se tomara la foto (Ilustración 11)

Se creó un repositorio en GitHub (<https://github.com/RonnyToledo/ESP32-CAM-PIR-BOT_TELEGRAM>) donde se incluyó el código fuente del programa, una explicación de su funcionamiento y un video donde se expone la aplicación ya terminada y funcional.

1. Discusión

* Evaluación en cuanto a utilidad:

El proyecto desarrollado presenta una gran variedad de aplicaciones ya que al ser un sistema de vigilancia basando en componentes baratos se puede implementar en más lugares proporcionando mayor seguridad a los entornos públicos y privados ya sea centros de trabajo, escuelas parques etc.

* Identificación de problemas o limitaciones:

Durante el proceso de desarrollo del programa nos encontramos con una limitante, la cual fue que a pesar de que todos los códigos funcionaban correctamente por separado a la hora de unirlos el código de inicializar y guardar imágenes en la tarjeta SD presentaba problemas los cuales no pudimos encontrar solución por lo que se decidió dejar el almacenamiento de imágenes solo en el bot de Telegram.

**Conclusiones**

El sistema de Monitoreo en tiempo real con ESP32-CAM desarrollado en este proyecto presenta varias ventajas como lo son su bajo costo de producción, la alta seguridad que brinda para sus compradores, la eficiencia a la hora de recibir las imágenes ya que estas se almacenan en Telegram, aplicación que posee servidores fijos por lo que el usuario podrá ver y compartir las imágenes en cualquier momento siendo así más rápida su respuesta ante una detección de movimiento sospechosa.

**Recomendaciones**

Se recomienda corregir solucionar el problema de la tarjeta SD de modo que se pueda contar así con una copia de seguridad de las imágenes en estado físico lo cual brindaría aún más seguridad al sistema.

**Referencias bibliográficas**

|  |  |
| --- | --- |
| [1] | «Secure Communication in IoT,» *IEEE Journal on Selected Areas in Communications.,* (2023. |
| [2] | Google, «Google Drive API Documentation. (2023),» (2023). |
| [3] | [En línea]. Available: https://www.330ohms.com/blogs/blog/crea-tu-propio-bot-de-telegram-con-esp32. |

HWLibre. (s.f.). *HC-SR501: Sensor de movimiento PIR*. HWLibre. Recuperado de [*https://www.hwlibre.com/hc-sr501/*](https://www.hwlibre.com/hc-sr501/)

Bitwise Ar. (2023, 11 de abril). ×ESP32 Telegram Bot Sensor PIR× [Video]. YouTube. [*https://youtu.be/WOb1QoqDoiI?si=ZziIHQJpO4Cav9du*](https://youtu.be/WOb1QoqDoiI?si=ZziIHQJpO4Cav9du)

*Copilot: https://www.bing.com/chat*

**Anexos**

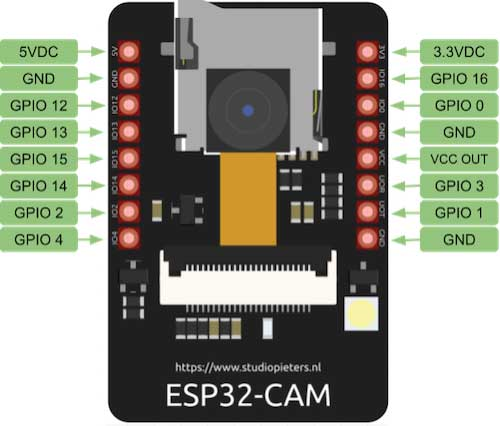
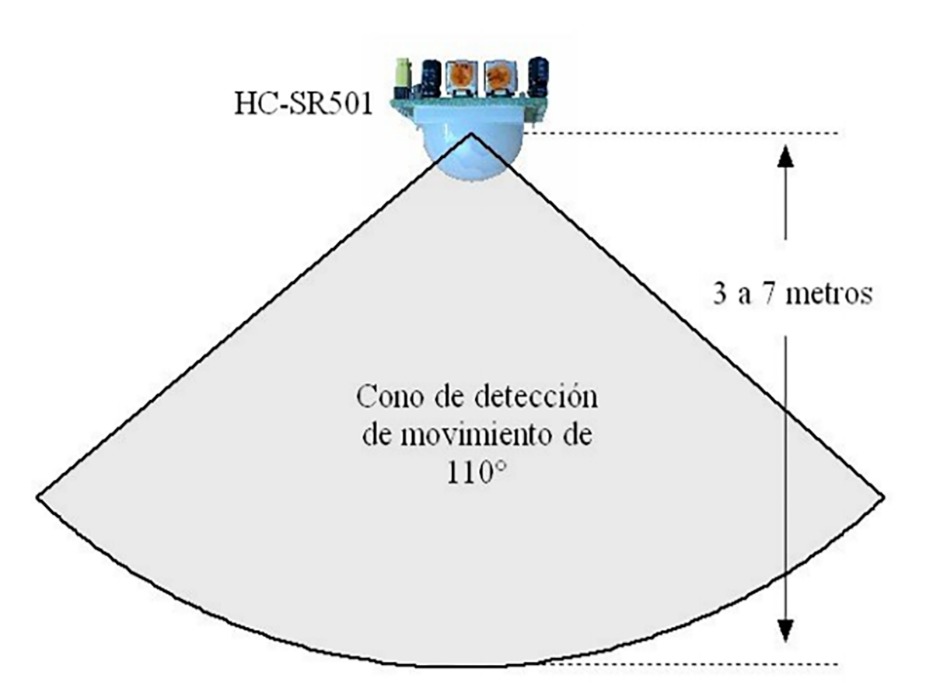
 

Ilustración 1 Ilustración 2

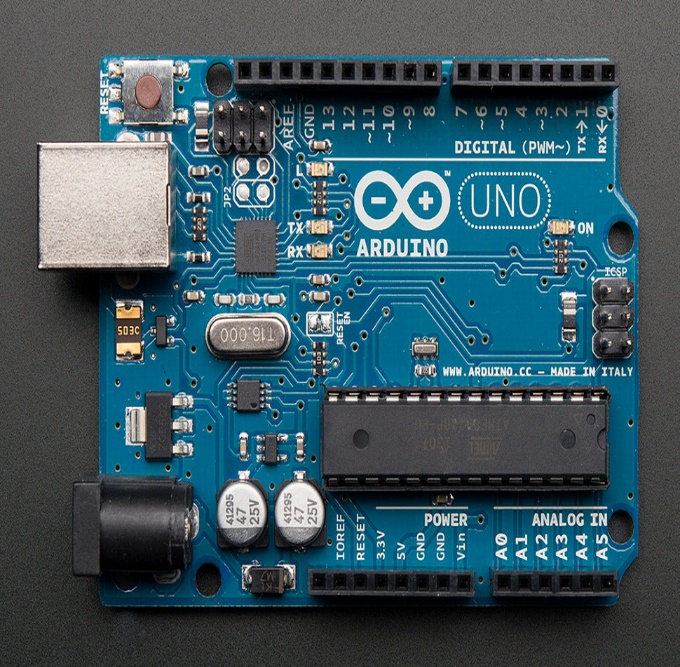
 

Ilustración 3 Ilustración 4

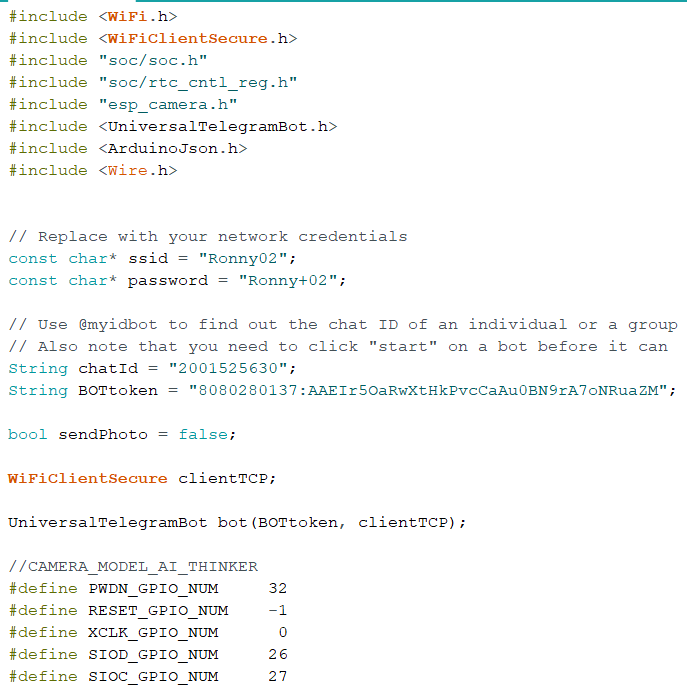
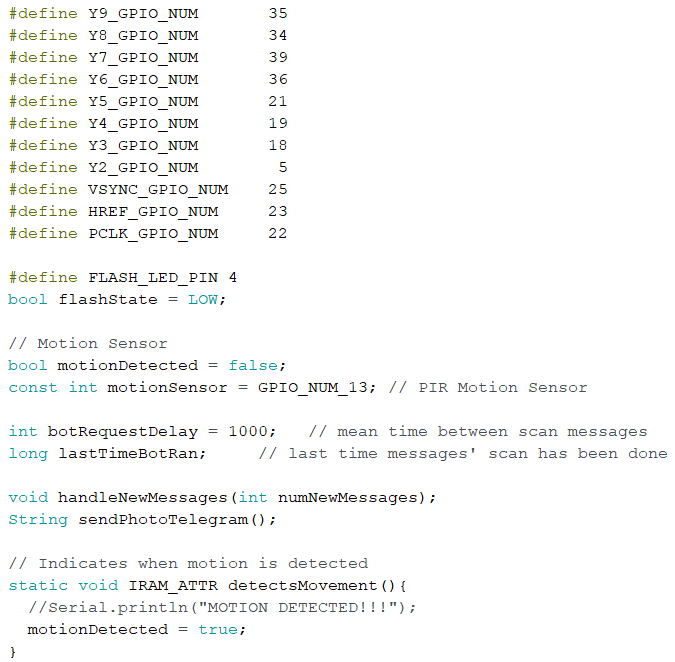
 

Ilustración 5

Ilustración 6

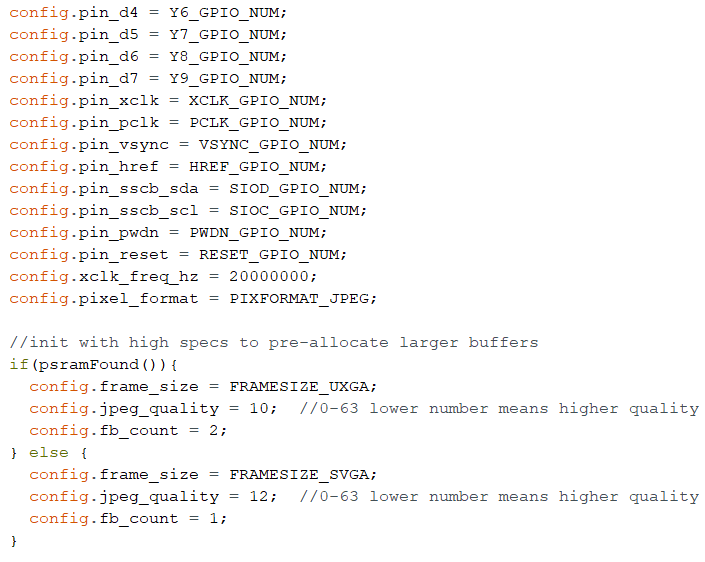
 

Ilustración 8

Ilustración 7

Ilustración 9

Ilustración 10

Ilustración 11 Ilustración 12

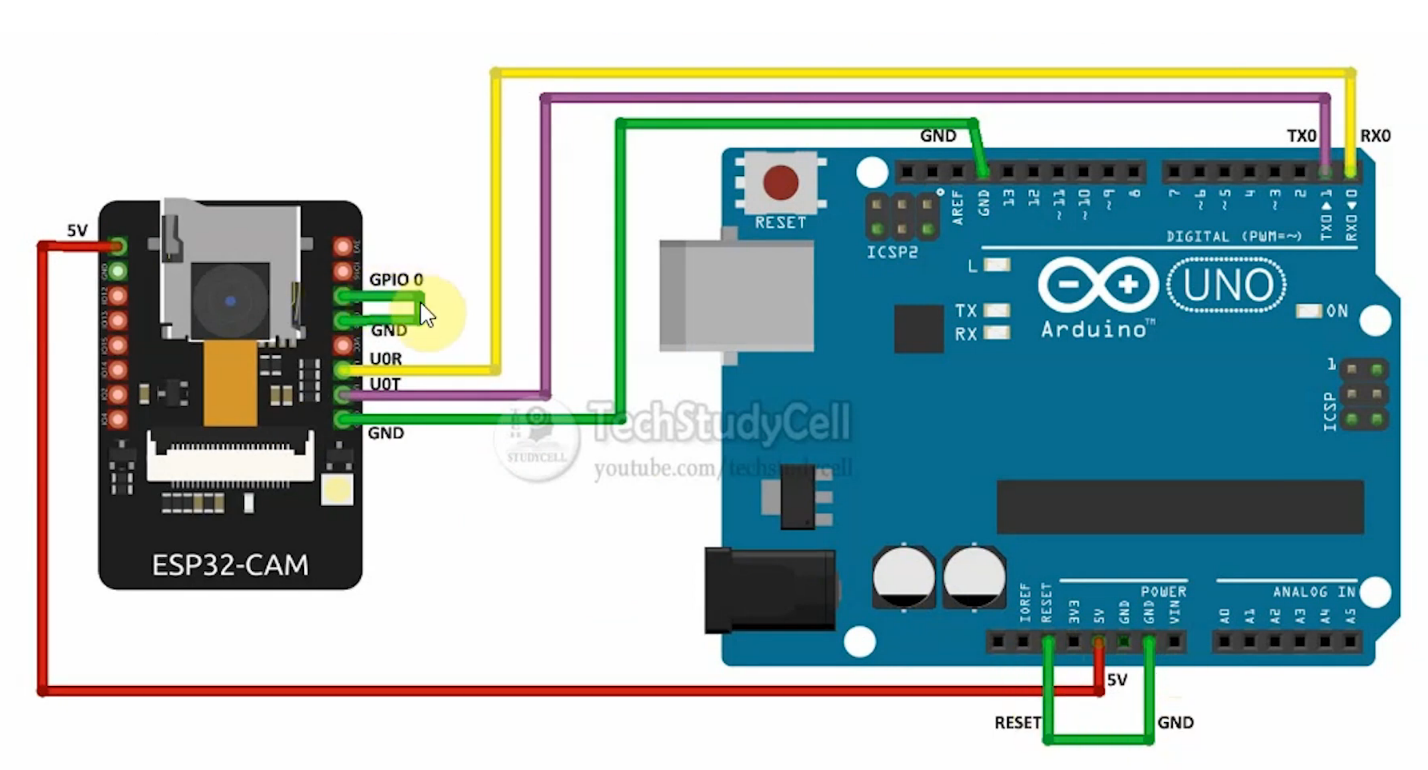


Ilustración 13



Ilustración 14



Ilustración 15



Ilustración 16