

Remote Control and Video Streaming via ESP32-CAM using Telegram Bot



Author: Aslan Bay Gush.

Abstract:

This project demonstrates a smart remote surveillance and control system using the **ESP32-CAM** microcontroller. Through integration with a **Telegram Bot**, user can:

- Turn the flash LED on/off remotely.
- Receive real-time photo from the camera.
- Get alerts upon motion detection (via PIR sensor).
- Retrieve the device IP address to access live video feed.

This system utilizes **Wi-Fi** connectivity, the **Telegram Bot API**, and Arduino IDE programming to create a secure and accessible interface via smartphone or computer.

Telegram:

Telegram is an instant messaging application that allows users to send text messages, photos, videos, files and animations to one another. It can be used simultaneously on multiple devices such as smartphones, tablets and personal computers.

Telegram also supports the creation of **bots**, which are small computer programs that can perform various tasks such as information retrieval, games, group management and more. Files of up 2GB per file can be shared, making Telegram a convenient platform for sharing large files.

Telegram Bot:

A **Telegram bot** is an automated computer program that can perform various tasks independently by receiving commands from users.

Telegram bots operate through an **API (Application Programming Interface)** and enable the creation of interactive experiences, service delivery, games, and more. using the app's messaging interface.

Bots function as Telegram users, but instead of sending messages manually, they receive and respond to messages through software code. Users can interact with bots by sending messages, clicking on custom buttons, or triggering commands.

API:



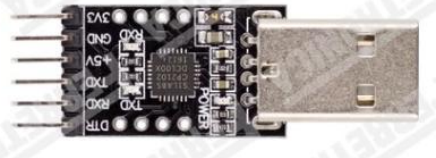

API (Application Programming Interface) is an interface that allows different software systems to communicate and connect with each other. An API provides a set of protocols, tools, and definitions that developers can use to create applications and connect to other services without needing to understand all the technical details behind the scenes.

How an API works:

The API acts as an intermediary between applications, receiving requests, processing them, and returning responses. The process involves three main stages:

1. **Request:** The application sends a request to the API, asking it to perform a specific action or retrieve data.
2. **Processing the request:** The API receives the request, performs the necessary processing (which may involve communication with other servers), and prepares the response.
3. **Response:** The API sends back the result of the request to the application.

Tool and Components:

Description	Components
Microcontroller with built-in camera	ESP32-CAM 
Motion detection	PIR Sensor (HC-SR501) 
USB-to-Serial for uploading code	FTDI CP2102  Or ESP32-CAM-MB 
Development environment	Arduino IDE
Arduino libraries	Arduino, esp_camera, WiFi, WiFiClientSecure,

	UniversalTelegramBot, ArduinoJson, esp_http_server
Wireless communication	Wi-Fi

How it works:

the ESP32-CAM connect to a Wi-Fi network.

A Telegram bot (created via [BotFather]) sends/receives commands.

The ESP32 receives:

/start → shows the options that the ESP32 can receive.

Options:

- /photo → sends back a photo.
- /flash → toggles the flash LED.
- /IP → sends back the device's IP.

The PIR sensor detects motion and triggers an automatic Telegram alert.

A browser can connect to **http:// [IP]** for live video.

Telegram Bot Setup:

We will create a simple bot using the **Telegram application**, while the actual commands will be written in the **Arduino software**.

The communication between the device and the application will take place through a **designated chat**.

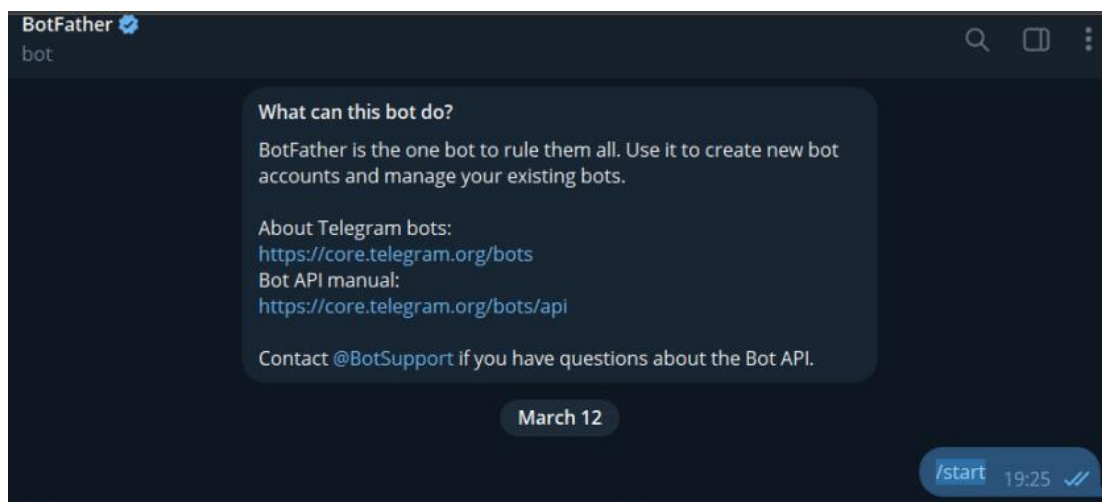
Step 1: Installing the Telegram Platform + Registration



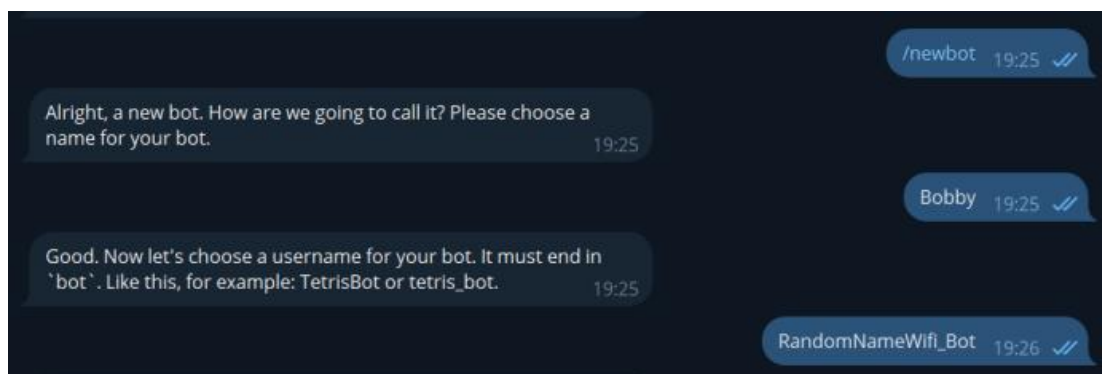
Step 2: Starting a Conversation with BotFather



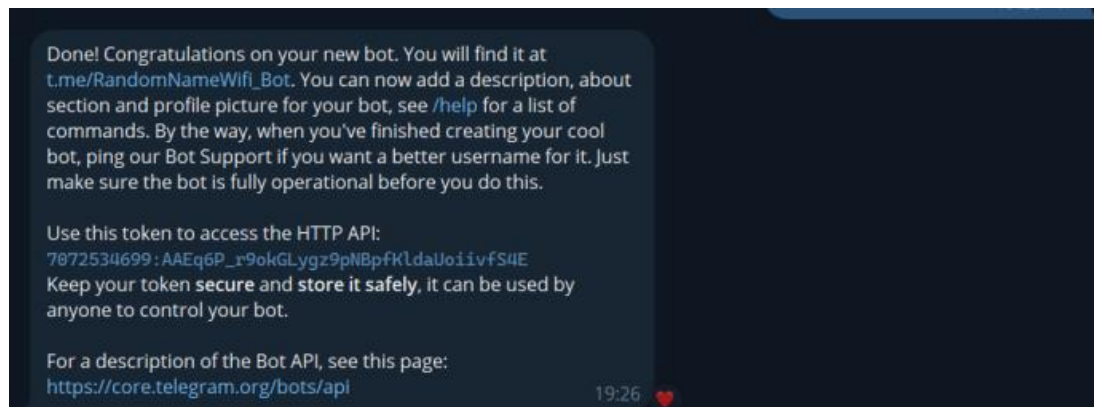
BotFather is an existing bot used to create new bots for your own use. You start by sending the command `/start`, followed by `/newbot`. Then, you can define a **name** and a **username** for your bot.



Step 3: Click the `/start` button and choose the `/newbot` option. Then, select a name and a username for your bot.



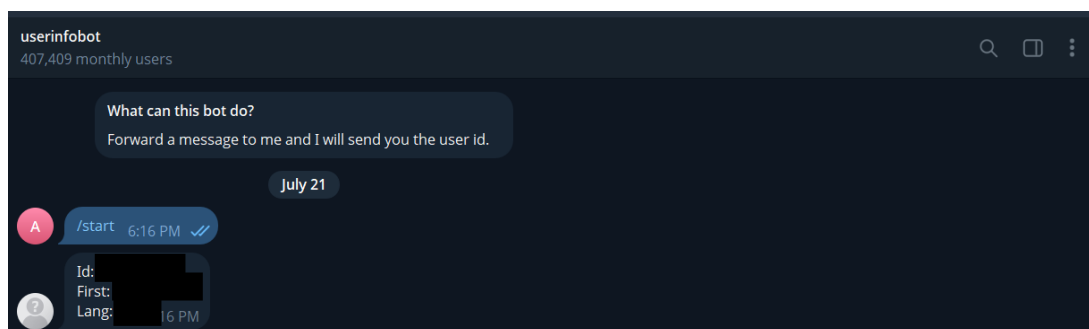
And then the BotFather send you a user token



Step 4: Obtaining the Chat ID

The Chat ID is a unique number assigned to each chat.

While it's possible to access our bot through a public link, to prevent unauthorized access, we can secure communication using the Chat ID. To retrieve it, start a conversation with `@userinfobot` — it will provide your Chat ID.

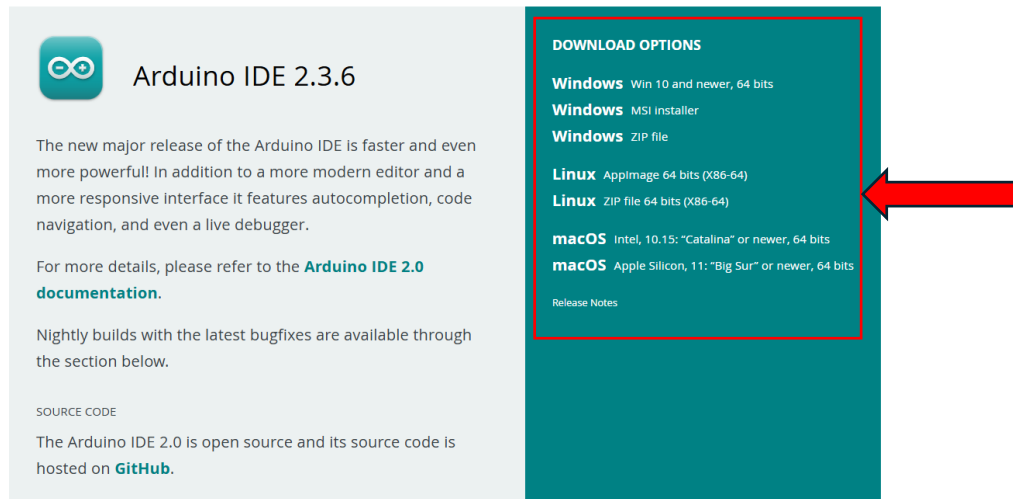


Arduino setup:

Step 1: Download the Arduino IDE:

<https://www.arduino.cc/en/software/>

Downloads



Step 2: After downloading and installing the Arduino IDE, we will install the libraries required to run our code.

You need to add the ESP32 URL to the Arduino IDE so it can find, download, and install the files needed to support ESP32 boards. Without it, the IDE doesn't know how to compile code or upload to an ESP32.

The URL tells the IDE where to get:

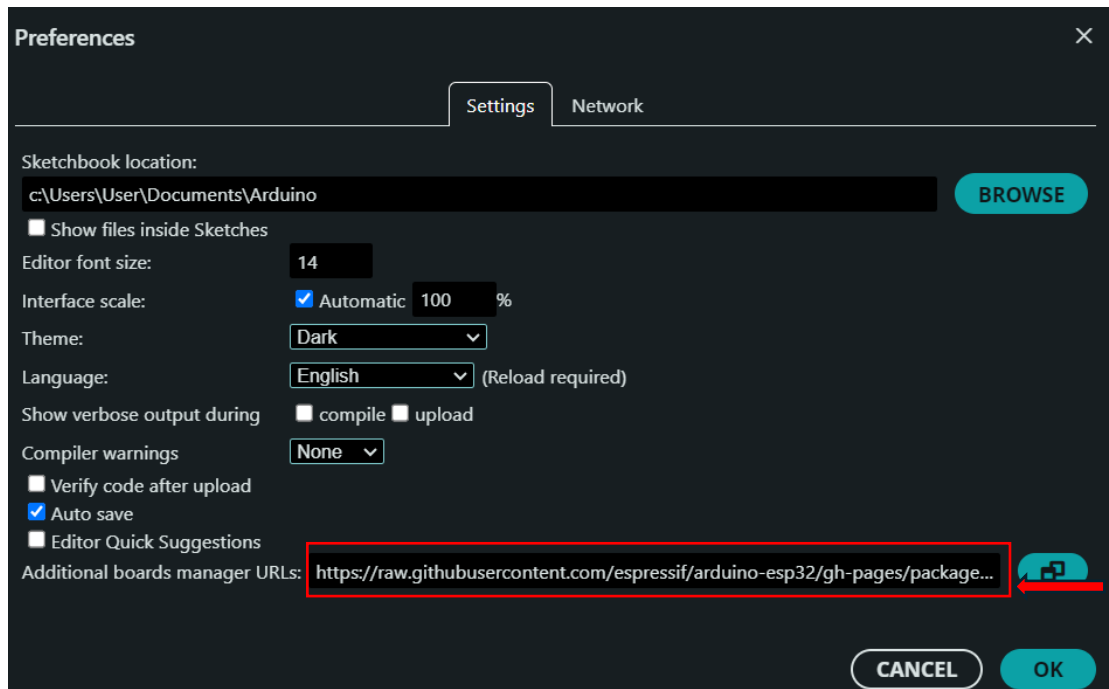
- Board definitions
- Compilers
- Upload tools

Without adding the URL, ESP32 won't appear in the Boards Manager and you can't program it.

Go to:

File → Preferences

Enter: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

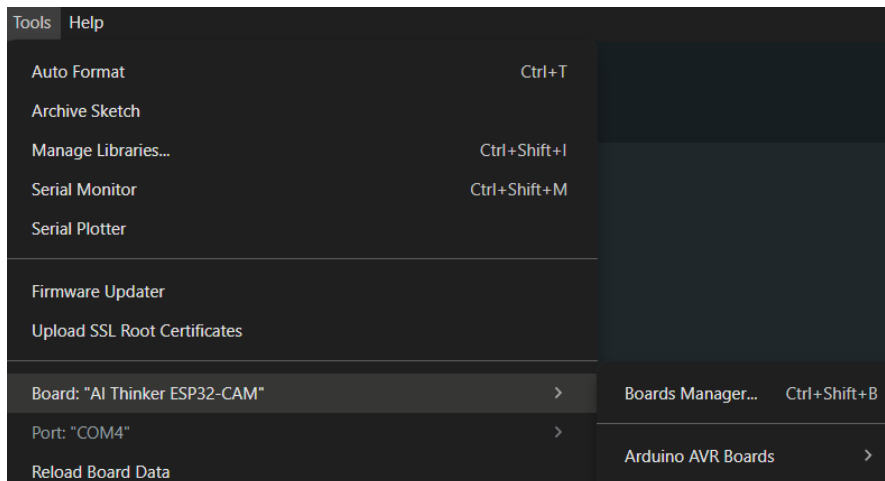


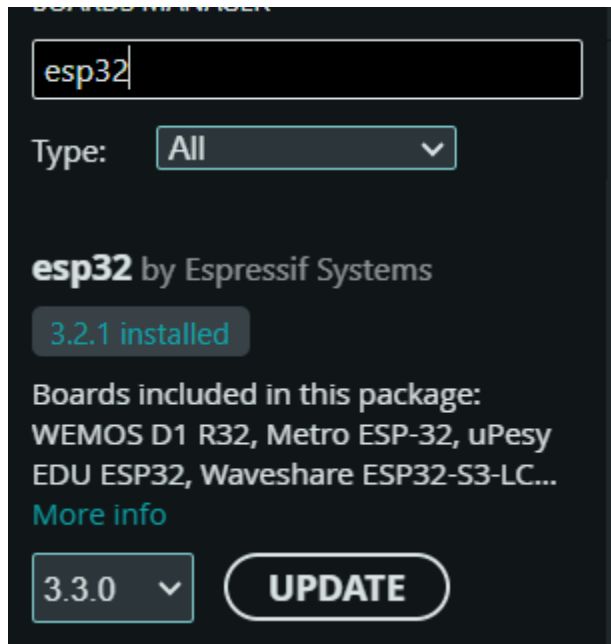
And press OK

Step 3: installing the esp32 boards.

Go to:

tools → board → board manger → esp32

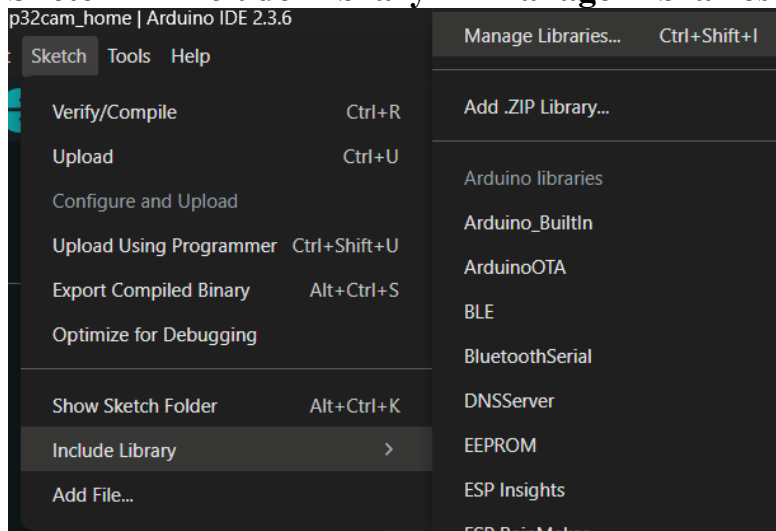




Step 4: installing the libraries.

Go to:

Sketch → Include Library → Manage Libraries



And install:

- Universaltelegrambot
- arduinoJson
- AsyncTCP
- ESPAsyncwebserver

Step 5: load the code **esp32-cam sensor telegram.ino** in the link below

[Link_code_github](#)

In the code:

`const char* ssid = "*****";` → enter the WiFi name.

`const char* password = "*****";` → enter the WiFi password.

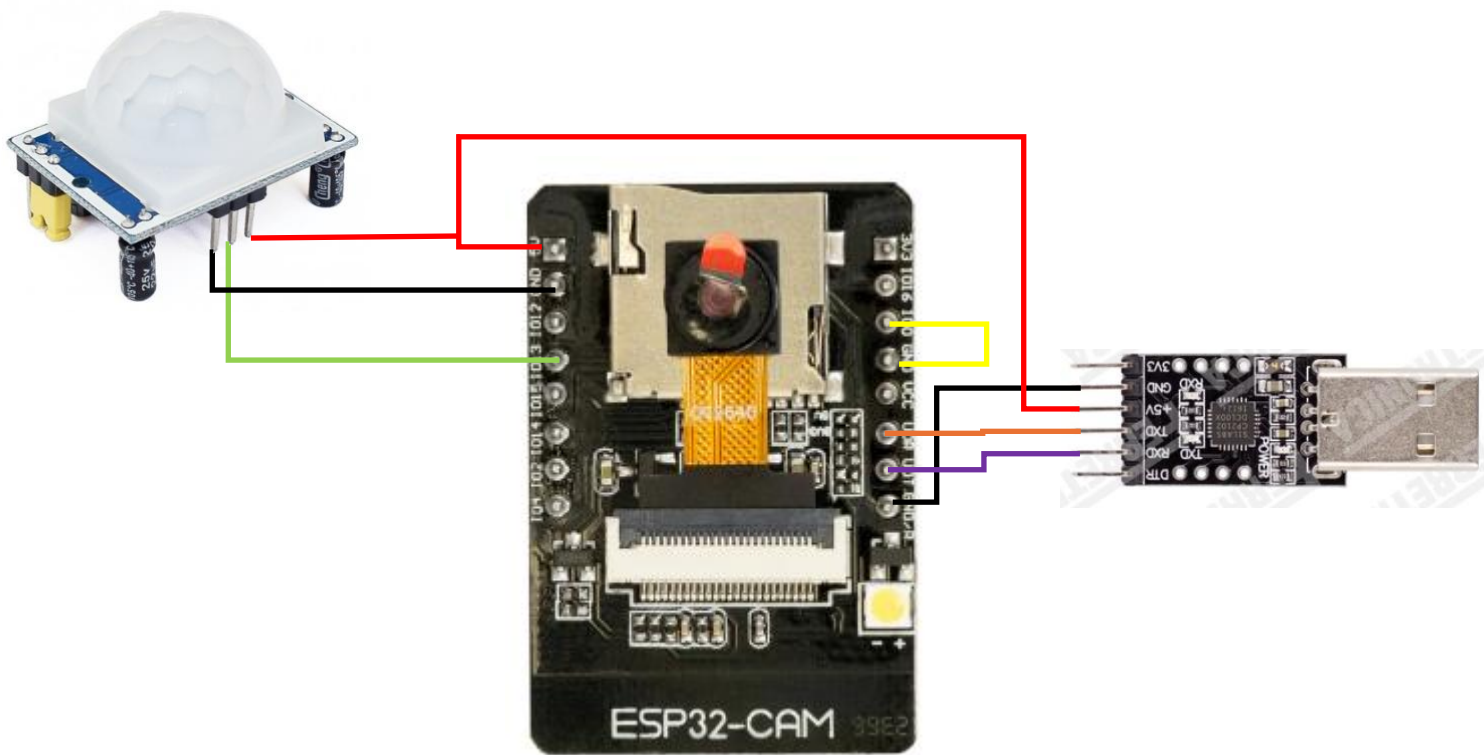
`String BOTtoken = "*****:****";` → enter the token from the telegram.

`String CHAT_ID = "*****";` → enter the ID from the telegram.

Hardware Assembly

Components and conactions:

- ESP32-CAM:
 - GND → pin IO.
- PIR Sensor (HC-SR501):
 - GND → GND (esp32-cam).
 - 5V → 5V (esp32-cam).
 - TXD → UOR (esp32-cam).
 - RXD → UOT (esp32-cam).
- FTDI Programmer (CP2102)
 - GND → GND (esp32-cam).
 - 5V → 5V (esp32-cam).
 - OUTPUT → pin IO13 (esp32-cam).
- Wires



Note!!!

After completing all the previous steps, compile and upload the code to the device using the Arduino IDE.

Make sure to connect GND and IO0 before uploading the code.

Once the upload is complete, disconnect IO0 and press the RESET button.

Then, open the Serial Monitor and wait for the Wi-Fi connection to establish.

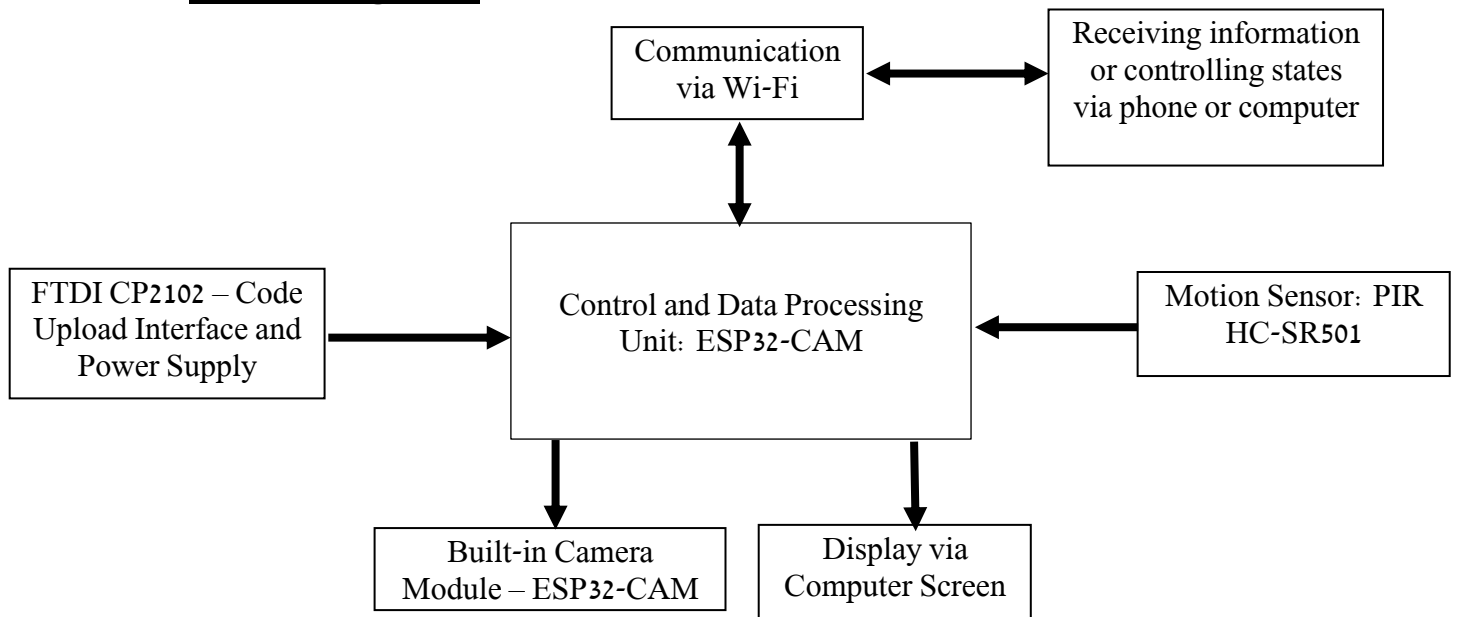
```
ets Jun  8 2016 00:22:57

rst:0x8 (TG1WDT_SYS_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:1448
load:0x40078000,len:14844
no 0 tail 12 room 4
load:0x40080400,len:4
load:0x40080404,len:3356
entry 0x4008059c

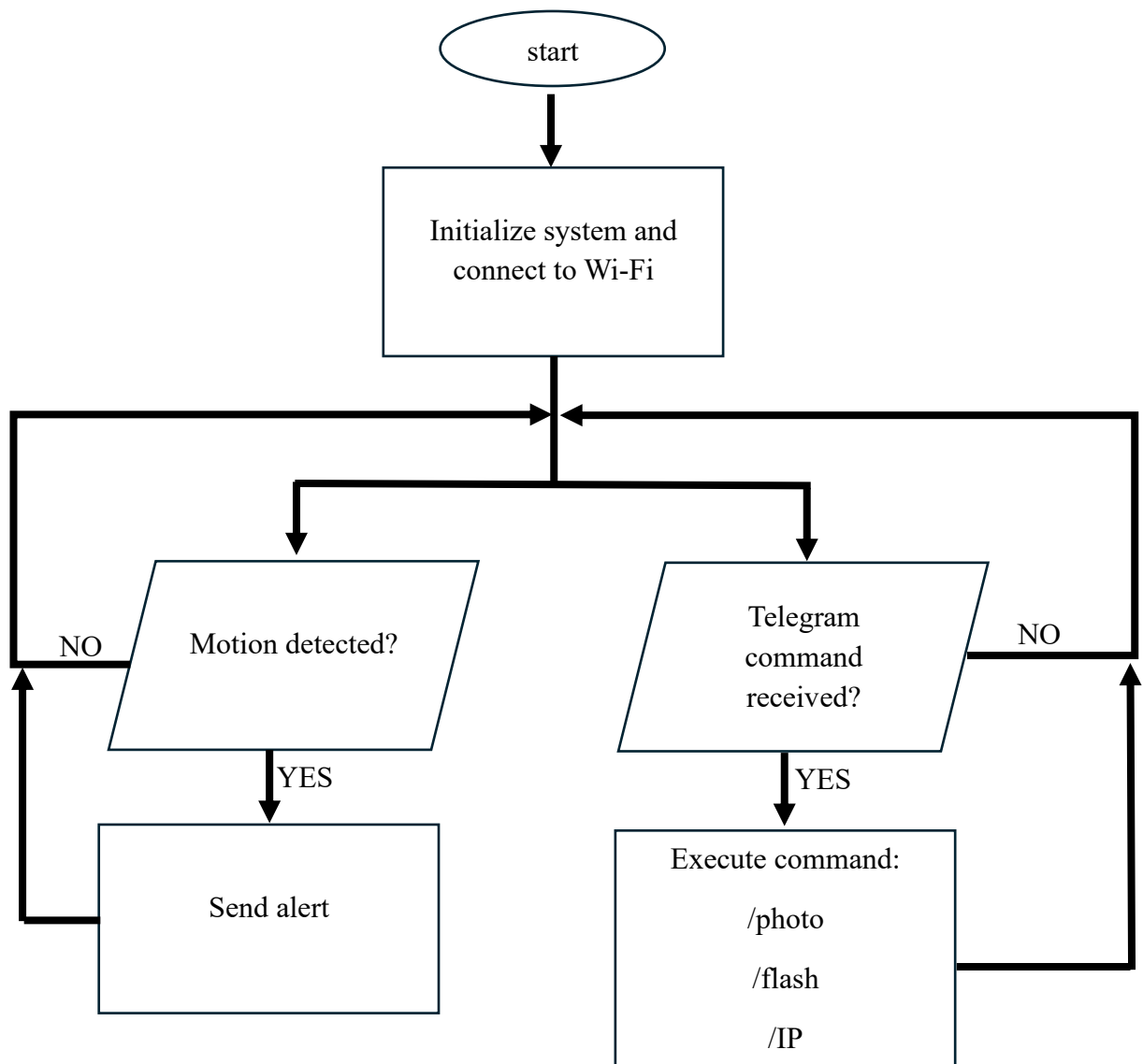
E (546) esp_core_dump_flash: No core dump partition found!
E (546) esp_core_dump_flash: No core dump partition found!

WiFi connected
Camera Ready! Use 'http://192.168.1.246' to connect
```

Block Diagram:



Flow chart:



Simulation:



