IoT-Based Motion Sensor Camera Surveillance System with Telegram Control Using ESP32-CAM

Author : Axay Ram
Email : axay19392@gmail.com
Designation : B.E Student at Vishwakarma Govt Engineering College
&
Author : Anurag Purohit
Email : er.apurohit@gmail.com
Designation : B.E Student at Vishwakarma Govt Engineering College

Vishwakarma Government Engineering College

Abstract:

In recent years, the demand for cost-effective, intelligent, and real-time surveillance systems has increased significantly due to the growing need for enhanced security in residential, commercial, and public spaces. This research presents the design and implementation of an **IoT-based Motion Sensor Camera Surveillance System** that leverages the **ESP32-CAM** module and **Telegram Bot** integration to provide a smart, user-friendly, and low-cost remote monitoring solution. The system combines the capabilities of the ESP32-CAM—a microcontroller with an integrated camera and Wi-Fi module—with a **PIR (Passive Infrared) motion sensor** to detect human presence. Upon detecting motion, the system automatically captures an image and sends it to a predefined Telegram account using a bot API, enabling instant alert and visual confirmation to the user regardless of their location.The system architecture is based on embedded IoT technology and real-time data communication via cloud-based messaging platforms. A user can remotely interact with the system using Telegram commands to request real-time images or control system behavior (e.g., turn the camera on/off, check system status). This two-way communication eliminates the need for a traditional monitoring station and provides flexible and secure access through smartphones.The project aims to improve home and office security through automation, ease of use, and affordability.

Introduction:

The rise of IoT technologies has enabled innovative solutions for remote monitoring and control. The ESP32-CAM, a low-cost microcontroller with an integrated camera, is an excellent platform for such applications. This project aims to create an IoT-based motion sensor camera system controllable via the Telegram app, initially conceptualized for operating a garage gate but adapted to focus on surveillance and light control. Key features include real-time photo capture, motion-triggered photography, light control, and environmental monitoring, all accessible globally through a WiFi-connected ESP32CAM and a Telegram bot. This paper details the system's design, implementation, and performance, highlighting its potential in home security and automation. Experimental results confirm the reliability and responsiveness of the system under various environmental conditions, showing minimal false triggers and consistent image transmission. This solution can be further extended to include features such as live video streaming, integration with cloud storage, and machine learning-based threat analysis, making it a promising prototype for next-generation smart security systems.

System Design:

The system combines hardware and software components to achieve its functionality:

1. Hardware Components:

- **ESP32-CAM**: Core microcontroller with a built-in OV2640 camera module.
- **PIR Sensor**: Connected to GPIO13 via a BC547 NPN transistor for motion detection.
- **DHT11 Sensor**: Connected to GPIO2 for temperature and humidity readings.
- **Relay Module**: Connected to GPIO12 to control an external light (or potentially a gate in future iterations).
- **Flash LED**: Connected to GPIO4 for illumination during photo capture.

Additional Components:

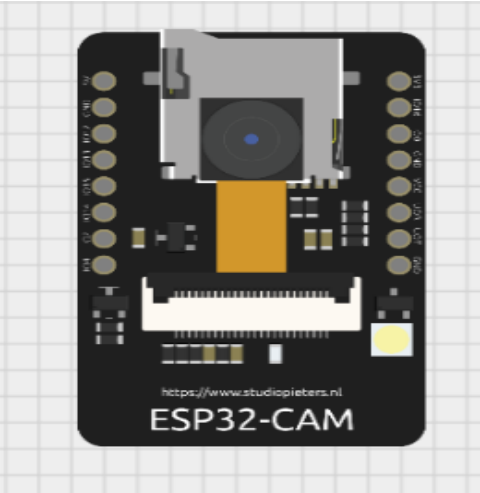
- 220-ohm and 1k resistors (2 each). o 1N4007 diode.
- BC547 NPN transistors (2). o 5mm LEDs (2). o 5V SPDT relay. o 2-pin terminal connectors (2).
- SPDT slide switch (across GPIO0 and GND for programming mode). o AC to DC converter (5M05 5V).
- FTDI232 TTL converter for programming.

2. Software Components

- **Arduino IDE**: Used to program the ESP32-CAM.
- **Telegram Bot API**: Facilitates command-based interaction.
- **Libraries**:
  - WiFi.h and WiFiClientSecure.h for network connectivity. o esp_camera.h for camera operations. o UniversalTelegramBot.h and ArduinoJson.h for bot communication. o DHT.h for sensor readings.

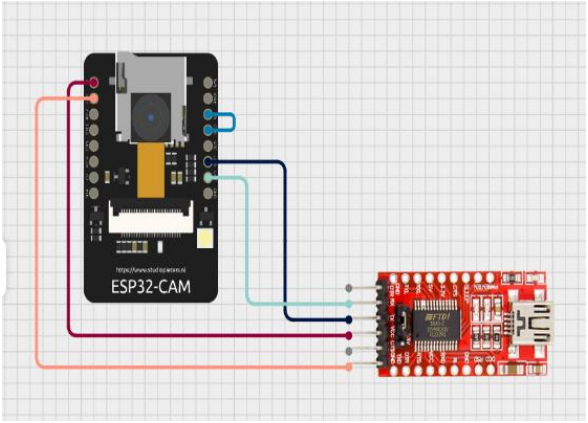
System Architecture:

1.ESP32CAM:



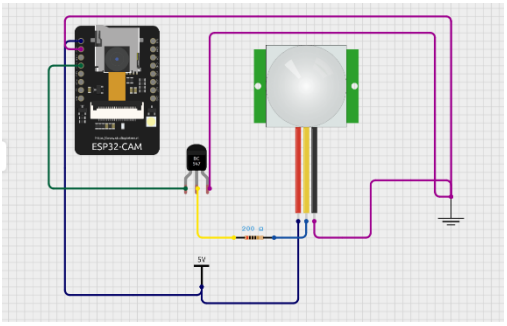
The ESP32-CAM module represents a balanced integration of powerful processing capabilities, versatile wireless connectivity, and high-resolution imaging in a compact form factor. Its design is well-suited for a range of applications from real-time surveillance and edge computing to educational projects, making it an attractive option for both commercial and hobbyist implementations. When integrating this module into a system, careful consideration must be given to power management, thermal performance, and network security to ensure optimal performance in the target application.

2.FTDI PROGRAMMER:



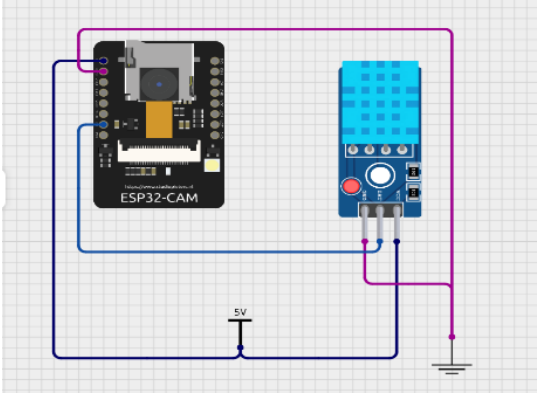
FTDI programmers play a critical role in the landscape of embedded system development and firmware programming. With their integrated USB-to-serial conversion capabilities, configurable voltage levels, and reliable performance, these devices facilitate seamless communication between a host computer and microcontrollers. Their applications span from initial development and debugging to mass production and field maintenance, making them indispensable tools in modern electronics engineering.

3.PIR SENSOR:



The PIR sensor plays a **crucial role in motion-triggered systems**, especially where power efficiency, cost, and reliability are important. When integrated with a powerful microcontroller like the ESP32-CAM, **it forms a complete and intelligent surveillance solution. Its simplicity,** low cost, and effectiveness make it a cornerstone component in modern embedded security systems.

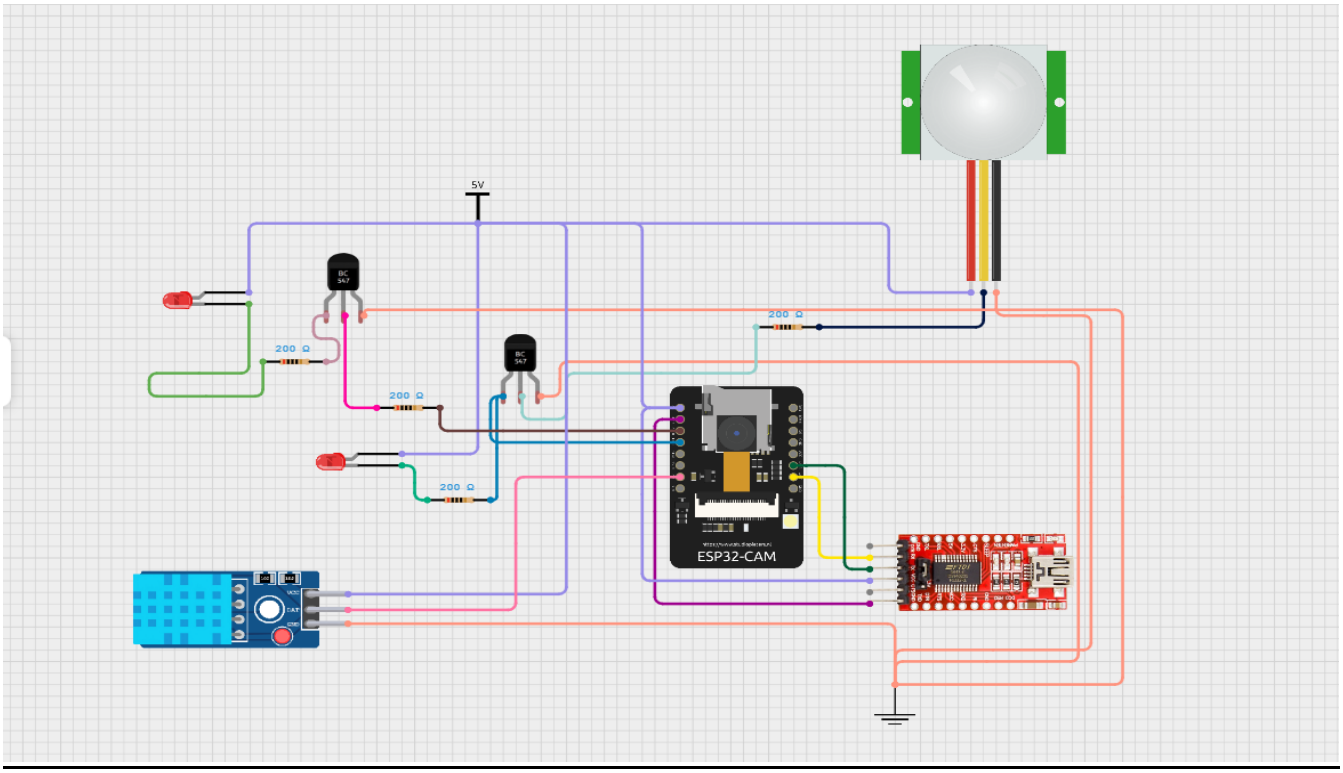
4.DHT11SENSOR:



The DHT11 sensor represents a balanced trade-off between cost, simplicity, and functionality. Although it has limitations in terms of range and measurement precision, it remains a popular choice for educational purposes and basic IoT applications. Its integration of capacitive humidity sensing and thermistor-based temperature measurement, along with a simple single-wire digital communication protocol, make it a versatile and accessible component in environmental monitoring systems.

5.CIRCUIT DIAGRAM:

The ESP32-CAM module with an FTDI programmer interface to enable easy firmware flashing and debugging through a USB-to-serial conversion, while sensor inputs such as the PIR module are connected to designated GPIO pins to detect motion and trigger actions; power is delivered via a regulated 5V supply that is stepped down to 3.3V for sensitive components, and decoupling capacitors are strategically placed to stabilize voltage levels, ensuring reliable operation. The camera sensor, along with an onboard or external antenna, facilitates wireless streaming and communication over Wi-Fi, and optional peripherals like a microSD card slot allow for local data logging. Overall, the circuit is designed to combine real-time video capture, motion detection, and seamless programming in an IoT framework, making it an efficient and versatile solution for applications such as smart surveillance, automated monitoring, and embedded system prototyping.



Advantages:

- **Versatility:** Integrates high-resolution imaging, motion detection, and remote programming in one compact module, which is ideal for a wide range of IoT applications.
- **Cost-Effectiveness:** Combines multiple functionalities into a single, compact design, reducing the need for separate components and lowering overall costs.
- **Ease of Integration:** Compatible with common development platforms (such as Arduino and ESP-IDF), offering extensive community support and simplified programming.
- **Efficient Power Management:** Designed to operate on regulated power supplies with decoupling capacitors in place, supporting stable operation even under variable power conditions.
- **Scalable Connectivity:** Provides built-in wireless communication (Wi-Fi and optional external antenna support) for seamless integration in networked and remote monitoring systems.

Disadvantages:

- **Resource Limitations:** The processing power and memory available on the ESP32-CAM might restrict performance when handling intensive tasks like simultaneous video streaming and sensor data processing.
- **Power Complexity:** Different modules require varying voltage levels (e.g., 3.3V for the ESP32 and 5V for other components), which complicates the power supply design and necessitates efficient regulation.
- **Connectivity Challenges:** Relying on an onboard PCB antenna may limit wireless range or performance in environments with high electromagnetic interference.
- **Design Complexity:** Integrating multiple subsystems (camera, sensor, and USB-to-serial programming) into one design increases circuit complexity, potentially leading to challenges in debugging and maintenance.
- **Limited Update Rate:** PIR sensors and certain modules may have constraints on data refresh rates, which can affect real-time performance if rapid detection or constant monitoring is essential.

Telegram Bot Creation:

1. Install the Telegram app from Google Play/App Store.
2. Search for "BotFather" (t.me/botfather), start it, and type /newbot.
3. Provide a unique bot name and username (ending in "bot").
4. Copy the generated bot token (e.g., 8159016295:AAG9m9MZbl91aypEOm0M5VvzO4IaVVO9MqQ).
5. Search for "myidbot" (t.me/myidbot), start it, and type /getid to get your chat ID (e.g., 5926504394).

Programming the ESP32-CAM:

1. **Arduino IDE Setup**:
    - Add ESP32 board support via Preferences > Additional Boards Manager URLs: https://raw.githubusercontent.com/espressif/arduino-esp32/ghpages/package_esp32_dev_index.json.
    - Install ESP32 board (version 2.0.5). o Install libraries: UniversalTelegramBot (1.3.0), ArduinoJson (6.20.0), DHT (1.4.4).

2. **Code Upload**:
    - Turn on the slide switch to connect GPIO0 to GND (programming mode). o Connect the FTDI232 to the ESP32-CAM as per the circuit.
    - Select Board: "AI Thinker ESP32-CAM", Partition Scheme: "Huge APP (3MB No OTA/1MB SPIFFS)", and appropriate COM port.
    - Upload the code below.
    - Turn off the switch and reset the ESP32-CAM to exit programming mode.

Results:

The system was tested with the following outcomes:

- **Photo Capture**: Successfully captured and sent photos via /photo and /photoWithFlash commands, with QVGA resolution.
- **Motion Detection**: Detected motion via PIR and sent photos without flash, with a 1-second delay to prevent rapid triggers.
- **Light Control**: Toggled the relay (light) using /lightOn and /lightOff.
- **Flash Control**: Toggled the flash LED independently with /flash.
- **Weather Monitoring**: Provided accurate temperature and humidity readings via /weather.

The system responded promptly, with satisfactory photo quality for surveillance purposes, making it viable for security applications.

Conclusion:

This IoT-based motion sensor camera system demonstrates the ESP32-CAM's potential for remote surveillance and automation. Controlled via Telegram, it offers photo capture, motion detectiolight control, and environmental monitoring, suitable for home security. Future iterations could expand its scope, making it a robust solution for realworld applications.

References:

- ESP32-CAM Documentation
- ESP32 CAM GUIDE
- ESP32 CAM PROJECT GUIDE
- Arduino IDE and Libraries (UniversalTelegramBot, ArduinoJson, DHT)