ReadMe – Final Project for the "Internet of Things (IoT)" Course

Project Title

Smart ESP32-CAM Surveillance System

Full Name(s) of Participant(s)

1. Sagibek Adil
2. Kenesbayev Arsen
3. Sagimbayev Nurbek

1. Project Overview

The **ESP32-CAM Surveillance System** is a compact and versatile monitoring device designed to observe and track human movement both automatically and manually. Utilizing the ESP32-CAM module, the system can detect and follow a person using computer vision or allow a user to control the camera's orientation with directional buttons (up, down, left, right). This dual-mode functionality makes it suitable for smart home security, remote monitoring, or DIY robotics projects, offering flexibility and ease of control in real-time.

2. Hardware Components Used

Arduino UNO, breadboard, servo and stepper motors, jumper wires, ESP32cam module, Power bank

3. Software Tools

- IDE: Visual studio and Arduino IDE
- Libraries:

FOR Manual:
ESP32 Arduino Core
ESP32 Camera Library
ESP32 Servo Library
ESP HTTP Server Library
ESP-DSP Library
ESP Timer Library
ESP GFX Library

FOR Automatic:
OpenCV (cv2)
NumPy (numpy)
Requests (requests)
Time (time)
Urllib (urllib.request)
Threading (threading)
OS (os)
Datetime (datetime)
Pyttsx3 (pyttsx3)
SpeechRecognition (speech_recognition)
Webbrowser (webbrowser)
Wikipedia (wikipedia)
PyWhatKit (pywhatkit)
Smtplib (smtplib)
Subprocess (subprocess)
Sys (sys)

- Data Visualization:
Not used

4. System Workflow

Manual Mode (Web Interface Control)

1. Connect the ESP32-CAM to a power source (e.g., power bank or USB adapter).
2. On your phone or PC, create a Wi-Fi hotspot with:
   - **SSID:** 11t
   - **Password:** 123456789
3. Wait for the ESP32-CAM to connect to the hotspot.
   - You can verify this by checking the list of connected devices on your phone. Note the IP address assigned to the ESP32-CAM.
4. Open a browser and go to the camera's IP address (e.g., http://192.168.143.186).
   - This will open the live video stream with directional controls.
5. Use the interface buttons to move the camera **up**, **down**, **left**, or **right**.
6. That's it — manual control is now active

Automatic Mode (Real-Time Tracking & Voice Control)

1. Power the ESP32-CAM using any USB power source.
2. Connect it to the same Wi-Fi network (SSID: 11t, Password: 123456789).
3. On your computer:
   - Open the folder containing the project files.
   - Open NewLook.py and **update the camera IP address** based on the one shown in your phone's connected devices.
4. Run the voice assistant system by executing the file gpt.py.
5. The system will begin listening for voice commands such as:
   - **"follow me" / "track me"**
   - **"follow and record"**
   - **"follow without recording"**
   - **"stop following" / "stop tracking"**
6. Based on your command:
   - The camera will follow the detected person using OpenCV-based tracking.
   - If "record" is included in the command, the video will be saved when you stop tracking.
7. Optional fun feature: You can also say:
   - **"play [video name] on YouTube"** (e.g., "play Perfect by Ed Sheeran")
     → This will open the YouTube video in your browser.
8. For best performance, ensure a **stable and fast internet connection**.

5. How to Launch the Project

Initial Setup

1. Connect the ESP32-CAM to a power source (via USB or power bank).

2. Create a mobile hotspot with:

   SSID: 11t
   Password: 123456789

3. Wait for the ESP32-CAM to connect to the hotspot

   - Check your phone's connected devices list and find the IP address of the ESP32-CAM.

🦾 Manual Control Mode

1. Open a browser and go to the ESP32-CAM's IP address (e.g., http://192.168.143.186).
2. Use the on-screen controls to manually adjust the camera's position.
3. Done! You now have live video and directional control.

🤖 Automatic Control Mode (Tracking + Voice Commands)

1. On your PC:
   - **First time only**: Open a terminal and install dependencies:
   - pip install -r requirements.txt
2. Make sure the camera's IP address is correctly updated in NewLook.py.
3. Run the tracking system:

   python gpt.py
4. Speak commands such as:
   - **"follow me"**, **"track me"** — to start following
   - **"follow and record"** — to track and save the video
   - **"follow without recording"**
   - **"stop following"**, **"stop tracking"** — to stop tracking and save recording (if enabled)
5. For YouTube feature (optional):
   - Say: **"play [video name] on YouTube"**
6. That's it — the camera now tracks and responds to your voice commands!

6. Implementation Highlights

What Worked Well

- ✅ **Manual Control via Web Interface**
  The live video streaming and directional control through the ESP32-CAM's local web server worked smoothly. The interface was responsive, and camera movement reacted well to user commands.
- ✅ **Wi-Fi Connection and Hotspot Setup**
  The ESP32-CAM consistently connected to the custom mobile hotspot ("11t") without issues, making setup easy and repeatable.
- ✅ **Real-Time Person Tracking with OpenCV**
  The Python script using OpenCV successfully detected and tracked a person in real time, maintaining smooth performance when lighting conditions were good.
- ✅ **Voice Command Integration**
  The system was able to recognize and respond to voice commands like "follow me" or "stop tracking," enhancing the user experience with hands-free control.

Challenges Faced & How They Were Solved

Fragmented Resources and Code Integration

➤ *Challenge:* One of the biggest difficulties was that there was no single complete guide or source for building a project like this. We had to explore multiple tutorials, GitHub repositories, and forums, each covering only part of what we needed. Combining code from different sources caused compatibility issues and required a lot of debugging and adaptation.

➤ *Solution:* We carefully studied each code snippet we found, tested them one by one, and gradually integrated them into our system. Although this process took time and effort, it helped us better understand how each component works — from the ESP32-CAM firmware to the Python tracking script and voice recognition module.

Attaching the ESP32-CAM to the Model

➤ *Challenge:* Physically mounting the ESP32-CAM onto the camera holder or model was more difficult than expected. We needed a secure but safe method that wouldn't risk damaging the board or causing a short circuit.
➤ *What Happened:* After trying various options, we ended up using a fabric-like rubber band to strap the ESP32-CAM in place.
➤ *Solution:* This surprisingly simple method worked well — the camera stayed stable, and since the material was non-conductive, there was no risk of short circuits or overheating. We just trusted the process and moved on!

Replacing Servo Motor with Stepper Motor

➤ *Challenge:* Replacing a servo motor with a stepper motor turned out to be more difficult than expected. We had to adapt the control code, rewire the connections, and physically mount the stepper motor in place. The mounting process was especially tricky.
➤ *What Happened:* Unfortunately, we accidentally broke one stepper motor while trying to glue it in place. We even used a soldering iron to melt the plastic for bonding — a desperate but somewhat effective solution.
➤ *Solution:* After trial and error, we managed to mount the motor and adjust the code accordingly so it would respond correctly to directional commands.

Finding the Right Power Supply

➤ *Challenge:* It was not easy to find the correct power source that would reliably run the ESP32-CAM and motors without overheating or damaging components.

➤ *What Happened:* At one point, we took a “leap of faith” and connected a 5V 2A phone charger to test the system — hoping it wouldn’t blow up.

➤ *Solution:* Luckily, it worked! Later, we found that using a power bank was a more stable and portable solution. It provided enough current and helped us avoid using unstable power adapters.

Voice Recognition Accuracy

➤ *Challenge:* In noisy environments or with unclear pronunciation, the voice assistant misinterpreted commands.

➤ *Solution:* We added feedback prompts and repeated the command recognition loop to improve accuracy. Using a better microphone also helped.

Synchronization between Systems

➤ *Challenge:* Ensuring the ESP32-CAM, tracking script, and voice control all stayed in sync was initially difficult.

➤ *Solution:* We structured the Python scripts to check camera availability first and gave users clear step-by-step startup instructions.

7. Demo Links (if available)

- Video: 🌐 esp32 AI camera with pan-tilt