**Prototype Planning**

IoT represents the collective network of devices connected, such as sensors and microcontrollers, along with enabling technologies like Artificial Intelligence and Cloud Computing that facilitate seamless communication of devices with the cloud or directly among devices themselves. This is a fast-emerging field of study that holds immense potential to bring a revolution in modern life by embedding technology into everyday objects. IoT allows for real-time monitoring, automation, and data-driven decision-making, changing the way we interact with technology and the environment. The rapid pace of digitalization has further empowered IoT to bring about solutions in areas such as healthcare, transportation, and security. Of these, one of the most important applications is security surveillance systems, which are necessary for the protection of homes, businesses, and public places. However, most of these solutions are either too expensive or not adaptable in dynamic scenarios. My project tries to address this gap by developing an IoT-based DIY security video streaming prototype. The system will integrate an ESP32-CAM module for live video streaming, an Arduino Uno microcontroller for motion detection using a PIR sensor, and a servo motor for dynamic tracking at a 180° field of view. This prototype, using simple and low-cost electronic components, shows that great functionalities can be achieved without necessarily going for higher and more expensive solutions. It really shows how creative design and effective use of resources can lead to exceptional results in the exciting domain of IoT. The proposed system detects motion instantly and sets the camera to focus towards the movement, maintaining that position for a few seconds, it also offers a dynamic view of the environment.

Live video streamed through onboard Wi-Fi modules and cameras on ESP32-CAM is viewed via a remote feed on any device connected on a network in a browser. It features a modular design that is highly scalable and adaptable to use cases ranging from home security and industrial monitoring to other applications such as DIY drones or resource monitoring systems for water and electricity consumption. Indeed, most of the components in the system can serve a wide range of projects with only minor adjustments, showcasing just how versatile and potential this can be. This flexibility underlines the broader applicability of IoT in the creation of customized, cost-efficient solutions to real-world challenges which makes it very compelling to developers and innovators alike.

**Section 1: Prototype Identification and Planning**

**Section 1.1: Literature Review on Prototype Identification**

IoT has definitely revolutionized the concept of security in real time through monitoring and automation, backed by insights that promise a far more dynamic and truly effective system compared to traditional approaches. Sensors, cameras, microcontrollers, artificial intelligence, and cloud services are rapidly making the IoT-based surveillance system smart, capable, adaptable, and economically viable. Some of the major contributions IoT has made to surveillance include remote monitoring and control of security systems for convenience and operational management. One of the most affordable IoT technologies in surveillance is the ESP32-CAM, a versatile microcontroller with an integrated camera featuring Wi-Fi capability and ease of use, it is an excellent choice for DIY security systems. The low cost of the ESP32-CAM allowed individuals and organizations to deploy highly functional surveillance systems without the heavy price tag of commercial solutions. Besides, this microcontroller can easily integrate with various sensors and components, offering a customizable platform for creating tailored security solutions. Passive Infrared sensors are the most commonly used sensors in IoT-based surveillance systems because of their energy efficiency, reliability, and ability to detect changes in infrared radiation due to motion. The PIR sensor is very ideal for motion detection. Thus, triggering events like turning cameras on, sending alerts, or other security devices like alarms or lights. These PIR sensors dramatically enhance the capabilities of other devices, such as cameras, in a surveillance system by offering a robust and comprehensive security solution. Modernization of surveillance systems involves using servo motors that precisely control the field of view of the camera. With the integration of a servo motor with the ESP32-CAM, the viewing angle of the camera would dynamically change to track movement, hence minimizing blind spots and increasing coverage. This is contrary to most of the traditional camera systems that are normally static, with a fixed field of view, which limits the area of coverage. Dynamic field-of-view capabilities, facilitated by servo motors, let cameras follow objects in real time to enhance the flexibility and effectiveness of the system. That means many users can manage to monitor broader areas with fewer cameras, which cuts down the cost greatly and makes the installation simpler. Despite so many advantages offered by commercial surveillance systems, including those by Hikvision and Neo-Dristi, the hefty prices keep many of these beyond the reach of small businesses and individuals in favor of affordable deals. Although they offer low-budget systems above 5,000 rupees and below 10,000 rupees they are static cameras that provide limited coverage, advanced systems with dynamic viewing capabilities are usually quite expensive. This price disparity has led many to work out some sort of DIY solution using IoT devices such as the ESP32-CAM. It provides an extremely customizable and flexible solution for a fraction of the price compared to most commercial products. DIY surveillance systems make it even easier to add more control in designing and implementing this system; hence, one can work out whatever specifications are kept in mind. For instance, ESP32 CAM can be integrated with servo motors and PIR sensors for developing a very low-cost yet highly efficient surveillance system. A peculiarity of such a DIY concept is the involvement of the servo motor, which gives it the ability to dynamically follow the motion in space in larger amplitudes, hence wide areas, in order to capture everything and ensure the least blind area. The dynamism of cameras means that only a few pieces will set up surveillance in a wide area which is a great cost-saving than setting up multiple static systems. In this prototype design the ESP32-CAM was initially supposed to handle the load of the camera and the motion detection as well as servo, but very soon it became clear that such a microcontroller couldn't bear the load of all tasks combined. So, the Arduino Uno was brought in to handle the servo motor and the PIR sensor, while the ESP32-CAM handled the camera and Wi-Fi connectivity. This setup allowed the workload to be distributed more evenly, hence improving system performance and responsiveness. Inclusion of Arduino Uno made the system modular, where individual components may be upgraded or replaced independently as new technologies become available. The modular design ensures the system remains adaptable and future-proof, with users able to integrate new cameras or sensors without extensive modifications. Also, its potential to upgrade and maintain components without restriction minimizes losses through downtime, besides extending the lifespan of the entire system. Also, the desire for maximum coverage and minimum blind spots I decided the implementation of dynamic field of view against static field of view implementation. With the camera enabled to detect motion, a single unit can achieve near-full coverage of a monitored area, thereby reducing the number of cameras required, as well as the complexity of installation. This dynamic system will always optimize security coverage and it will be cost-effective with very high efficiency. It was concluded that the application of IoT technologies such as the ESP32-CAM, PIR sensors, and servo motors to DIY surveillance systems makes an alternative for most commercial ones as it is pretty affordable, flexible, and easily scalable. In turn, the use of these systems contributes to lowering costs and adding functionality while allowing for easy adoption of new technologies. This will allow a surveillance system with low-cost components and modular design to be effectively set up for specific needs while maintaining high security coverage and also will have room for future enhancements or improvements based on user preferences.

**Section 1.2: Reflection on Prototype Identification**

The research phase of the project provided essential insights into the strengths and weaknesses of various IoT components, guiding the selection process for the final prototype. The ESP32-CAM stood out as an excellent choice for video streaming due to its compact size, built-in Wi-Fi, and the OV2640 camera module, which offered seamless connectivity for the intended use case. In contrast, the PIR sensor was chosen for its reliable motion detection capabilities, particularly when paired with the Arduino Uno, making it a clear and straightforward selection. The more challenging decision came when choosing between a 180° and 360° servo motor. After careful consideration, I opted for the 180° servo motor, as it provided a practical solution for most use cases, offering sufficient coverage while minimizing unnecessary complexity. One of the most significant challenges I faced was balancing cost and performance, which I resolved by carefully selecting components that met the project’s requirements without exceeding the budget. This balancing act was crucial in maintaining system performance while keeping the prototype affordable. Through hands-on experimentation, I was able to gain a deeper understanding of the real-world challenges involved in integrating hardware with software. It was during this phase that the importance of modular design became particularly evident. Working with smaller, manageable sections of the prototype made debugging and enhancing individual components much easier, which ultimately contributed to smoother development. Furthermore, the modular design allowed for flexibility, ensuring that each component could be updated or replaced independently in the future, thus paving the way for scalability and further improvements. This careful planning and modular approach laid a solid foundation for the development phase, ensuring that the project objectives were clear and achievable, while providing a roadmap for future enhancements. To further enhance the system’s functionality, future developments could focus on incorporating AI-driven motion detection, improving video feed quality, and integrating cloud storage for seamless data access and backup.

**Section 2: Prototype Development**

Once all the components were gathered, I began by thoroughly testing each one to ensure there were no errors and to evaluate their precision and overall quality. This step was crucial to verify that each component functioned as expected before moving forward. After confirming the reliability of the individual parts, I proceeded to implement the prototype according to the plan I had developed during the research and design phase. This involved assembling the components and translating the initial ideas into a working model, ensuring that all elements were integrated effectively and functioned together as intended.

**Section 2.1: System Development, Developed Code, and Planning Documents**

The prototype consists of four components:

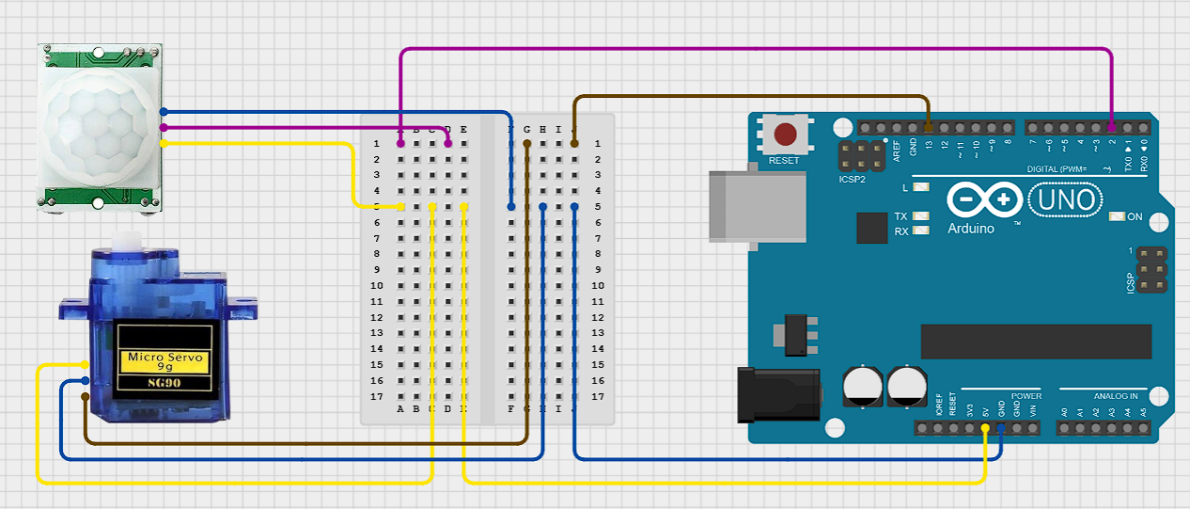
* ESP-32-CAM: streams live video by creating a web server after establishing connection with Wi-Fi using Wi-Fi module and the stream can be viewed from any device on the network simply by entering the URL.
* Arduino: This microcontroller is used to control both PIR sensor and servo motor.
* PIR sensor: It is used to detect motion.
* Servo Motor (180°): It is used to move entire setup 180° and provide dynamic field of view.

The codebase is divided into two modules:

* Arduino Code: Handles motion detection and servo control. The PIR sensor triggers the servo to focus on the detected motion for a while otherwise, the servo keeps moving from 0° to 180° with an increment and from 180° to 0° with decrement providing a dynamic field of view.
* #include <Servo.h>
* Servo myServo;
* const int pirPin = 2;
* int motionDetected = 0;
* int currentAngle = 0;
* bool increasing = true;
* void setup() {
* myServo.attach(9);
* pinMode(pirPin, INPUT);
* }
* void loop() {
* motionDetected = digitalRead(pirPin);
* if (motionDetected == HIGH) {
* delay(3000);
* } else {
* if (increasing) {
* currentAngle++;
* if (currentAngle >= 180) {
* currentAngle = 180;
* increasing = false;
* }
* } else {
* currentAngle--;
* if (currentAngle <= 0) {
* currentAngle = 0;
* increasing = true;
* }
* }
* myServo.write(currentAngle);
* delay(15);
* }
* }
* ESP32-CAM Code: Configures the camera for video streaming and establishes a Wi-Fi connection to start a webserver. (The code for ESP32-CAM is divided into multiple files which are sec\_cam.ino, app\_httpd.cpp, camera\_pins.h and camera\_index.h. So, I’ll just add main file i.e. sec\_cam.ino) The stream can be viewed in a custom web page just by entering the URL.
* #include "esp\_camera.h"
* #include <WiFi.h>
* #include "camera\_pins.h"
* const char \*ssid = "SSID";
* const char \*password = "PASSWORD";
* void startCameraServer();
* void setup() {
* Serial.begin(115200);
* Serial.println();
* camera\_config\_t config;
* config.ledc\_channel = LEDC\_CHANNEL\_0;
* config.ledc\_timer = LEDC\_TIMER\_0;
* config.pin\_d0 = Y2\_GPIO\_NUM;
* config.pin\_d1 = Y3\_GPIO\_NUM;
* config.pin\_d2 = Y4\_GPIO\_NUM;
* config.pin\_d3 = Y5\_GPIO\_NUM;
* config.pin\_d4 = Y6\_GPIO\_NUM;
* config.pin\_d5 = Y7\_GPIO\_NUM;
* config.pin\_d6 = Y8\_GPIO\_NUM;
* config.pin\_d7 = Y9\_GPIO\_NUM;
* config.pin\_xclk = XCLK\_GPIO\_NUM;
* config.pin\_pclk = PCLK\_GPIO\_NUM;
* config.pin\_vsync = VSYNC\_GPIO\_NUM;
* config.pin\_href = HREF\_GPIO\_NUM;
* config.pin\_sccb\_sda = SIOD\_GPIO\_NUM;
* config.pin\_sccb\_scl = SIOC\_GPIO\_NUM;
* config.pin\_pwdn = PWDN\_GPIO\_NUM;
* config.pin\_reset = RESET\_GPIO\_NUM;
* config.xclk\_freq\_hz = 20000000;
* config.frame\_size = FRAMESIZE\_UXGA;
* config.pixel\_format = PIXFORMAT\_JPEG;
* config.grab\_mode = CAMERA\_GRAB\_WHEN\_EMPTY;
* config.fb\_location = CAMERA\_FB\_IN\_PSRAM;
* config.jpeg\_quality = 12;
* config.fb\_count = 1;
* if (psramFound()) {
* config.jpeg\_quality = 10;
* config.fb\_count = 2;
* config.grab\_mode = CAMERA\_GRAB\_LATEST;
* } else {
* config.frame\_size = FRAMESIZE\_SVGA;
* config.fb\_location = CAMERA\_FB\_IN\_DRAM;
* }
* esp\_err\_t err = esp\_camera\_init(&config);
* if (err != ESP\_OK) {
* Serial.printf("Camera init failed with error 0x%x", err);
* return;
* }
* #if defined(LED\_GPIO\_NUM)
* setupLedFlash(LED\_GPIO\_NUM);
* #endif
* WiFi.begin(ssid, password);
* while (WiFi.status() != WL\_CONNECTED) {
* delay(500);
* Serial.print(".");
* }
* Serial.println("\nWiFi connected");
* startCameraServer();
* Serial.print("Camera Ready! Use 'http://");
* Serial.print(WiFi.localIP());
* Serial.println("' to connect");
* }
* void loop() {
* delay(10000);
* }

Planning Documents:

Two different mechanisms that make up this system are ESP32-CAM + (optional: ESP32-CAM-MB for programming and powering the ESP32-CAM as it doesn’t have an inbuilt USB port) and Arduino Uno + PIR sensor + Servo motor. These two different mechanisms are put together in a single board to make the final proposed system. Let’s see the wired connections of the system mechanisms and flowchart demonstrating data flow in the system mechanisms:



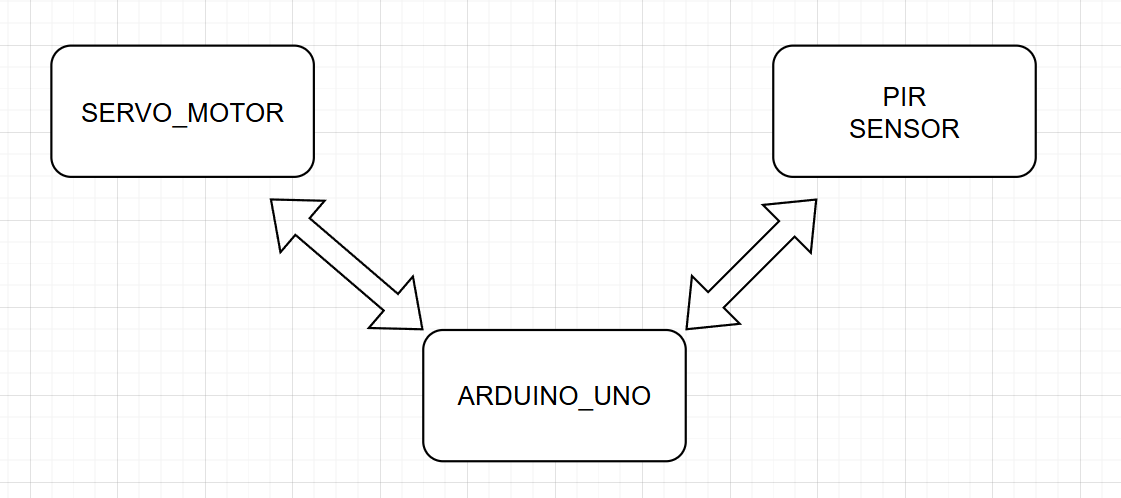
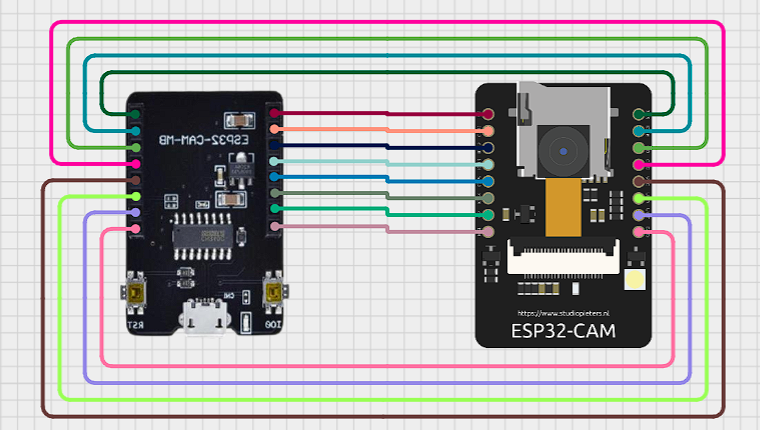


FIG: First mechanism for dynamic field of view & motion Detection.

Once the code is uploaded to Arduino Uno microcontroller, it reads signal from PIR sensor to detect motion. When no motion is detected, it moves the servo back and forth from 0° to 180° and vice versa. The servo's movement is controlled by gradually changing the angle in either direction, depending on the current state (increasing flag). If motion is detected by PIR sensor, the microcontroller receives signal and sends signal to servo which causes servo to pause for three seconds before continuing to move. This allows the second mechanism to provide dynamic field of view along with an opportunity to focus on object when motion is detected for better view of object. The Arduino Uno microcontroller can be reduced by removing it completely and connecting other components to ESP32 but the performance gets compromised so, I used a dedicated microcontroller just to overcome that but it isn’t really necessary.



The ESP32-CAM board is simply placed on top of ESP-32-CAM-MB board and connected with the inbuilt pins. This above diagram simply represents the wired connection between every pin of both the board for example 5v to 5v. (Note : The MB-board is upside down.)

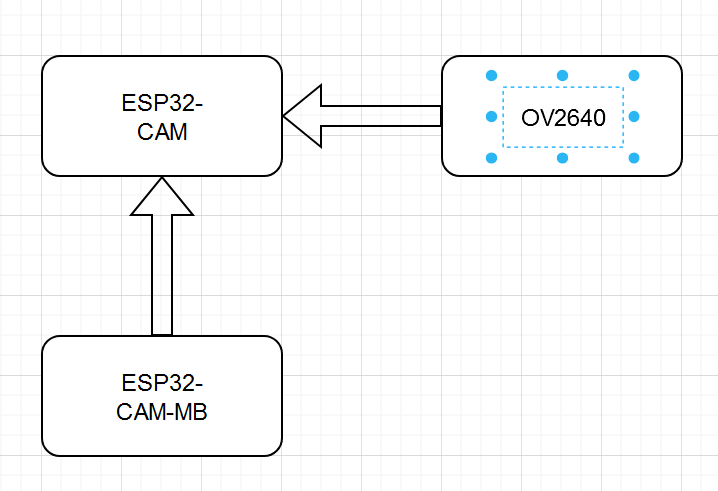


FIG: Second mechanism for video-streaming

(Note: OV2640 is connected to ESP32\_CAM using flexible flat cable inbuilt to OV2640 camera.)

Once the code is uploaded to microcontroller it configures an ESP32 camera module, initializes it, and sets up the necessary pins and frame settings. The ESP32 then connects to a Wi-Fi network using the provided SSID and password. The code starts a server (via startCameraServer()) that will stream live videos which can be accessed remotely by using URL provided in a web browser of a device connected to the network. Once we enter the provided URL we’ll see a custom page with start stream button which when clicked redirects to URL/stream and live video feed on a custom web page can be seen.

I combine these two different mechanisms to make a dynamic video streaming system. A board containing both PIR sensor and ESP32\_CAM on front and Arduino on back is placed on the propeller of servo which then moves around providing dynamic view. The PIR sensor and ESP32-CAM are placed on front so that they can work together even though they aren’t connected to each other. The Arduino Uno is placed at back to make the system look clean and the servo is placed at bottom with propeller connected to the board holding all other components that provides dynamic movement allowing 180° field of view and the servo motor can provide timely pause when a motion is detected.

A diagram of a computer

Description automatically generated

FIG: Basic diagram to represent system setup

Begin by ensuring the necessary code is uploaded to the respective microcontrollers: the ESP32-CAM with the OV2640 camera module and the Arduino Uno with the PIR sensor and servo motor. Follow the provided wiring diagram meticulously to ensure all connections are accurate and secure.

Once the connections are complete, supply power to both mechanisms of the system. Powering the ESP32-CAM module initiates the web server and starts the video streaming process, while the Arduino Uno powers the PIR sensor and servo motor, enabling motion detection and a dynamic field of view for the system. Ensure that the power source used is stable and reliable to avoid interruptions during operation.

The live video feed can be accessed and monitored using any device on the same network through the specified URL (e.g., <http://192.168.197.1/>). Open this URL in a web browser to navigate to a page with a "Start" button. Clicking this button redirects the user to the /stream page, where the live feed from the ESP32-CAM can be viewed.

For optimal performance, verify the following before starting the system:

* Use power cables that match the required configuration and specifications of the components to prevent power issues.
* Ensure the Wi-Fi network providing internet is reliable, fast, and stable, as interruptions in connectivity can hinder the streaming process.
* Confirm all components are securely connected, and the system is free of loose wiring or hardware faults.

Lastly, regularly monitor the system's performance during operation. In case of any issues, troubleshoot the wiring, power supply, or network configuration as needed to maintain uninterrupted functionality.

**Section 3: Evaluation**

**Section 3.1: Report on the Evaluation**

Initially, I started by thoroughly evaluating the performance of each individual component to ensure the system’s reliability and to identify any potential issues. By testing the PIR sensor, I was able to adjust its distance and timing settings to suit my needs through its dial, allowing me to optimize its motion detection. The servo motor’s movement was also fine-tuned for smoother operation, ensuring precise positioning. For the ESP32-CAM, I adjusted settings such as frames, buffer sizes, and video quality, optimizing it for consistent performance. Following the individual tests, I assembled the components into their respective mechanisms—first and second—integrating them into a functional prototype. During this assembly phase, I had to make small code adjustments, including modifications to the angle increment or decrement depending on the situation, delay times, and stream handling for web pages. After completing the assembly, I tested the full system, evaluating the combined performance of the PIR sensor, servo motor, and ESP32-CAM under real-world conditions. The system’s overall functionality was tested by monitoring the PIR sensor’s motion detection accuracy, the servo motor’s responsiveness, and the ESP32-CAM’s ability to maintain a stable video stream. The PIR sensor, even with the slight reduction in expected range (6 meters instead of 7), performed effectively with minimal false positives. As for the ESP32-CAM, its video streaming capabilities were adequate for basic security purposes, provided the device had a stable power source and internet connection. While the streaming was smooth with low latency, I noted that for more critical applications like wildlife monitoring or industrial settings, a higher-resolution camera like the OV3660 or OV5640 would be preferable. The servo motor operated smoothly without jitter, and it accurately adjusted its position based on the PIR sensor’s detected motion. The 180-degree servo motor was sufficient for this design, but for a 360-degree view, I would recommend using a servo motor capable of full rotation. Additionally, integrating cloud services would allow for cloud storage of video feeds, though I opted to record the video locally by capturing the stream on a separate device. This approach helped reduce the load on the ESP32, enhancing the overall performance of the system. Alternatively, using an SD card (such as a 4GB card, which is supported by the ESP32) could serve as a storage solution for recorded footage. A smart feature I integrated was storing photos only when motion was detected, which helps save storage space while still retaining useful footage. This motion-triggered photo storage mechanism further optimized the system’s storage management.

In terms of design, I decided on using two separate mechanisms, each serving a specific function. One mechanism aimed to reduce the load on the ESP32, which in turn improved the performance of video streaming. The second mechanism was focused on modularity, allowing the flexibility to update or replace individual components without affecting the rest of the system. For instance, if I needed to upgrade the video quality, I could simply swap out the camera for a better one, without needing to modify the entire system. This modular approach not only offered convenience in terms of upgrades but also allowed for easy experimentation with different sensors or components. Moreover, this setup provided an opportunity to implement future improvements such as AI integration for smarter motion detection and edge processing to reduce the false triggers from the PIR sensor. To further enhance system performance and reduce costs, I considered the option of removing the PIR sensor and the Arduino Uno microcontroller. Instead, I could implement object tracking directly through the ESP32 if the video feed quality was not a primary concern, or if component availability or costs became an issue. By removing the microcontroller, I would reduce the complexity of the design and lower the overall system cost. Additionally, I kept the project’s code modular by separating it into different files, which made it easier to adapt when upgrading or changing components. For example, if I decided to switch to a different camera module, I would only need to modify the camera configuration in the header file (camera\_pins.h) rather than rewriting the entire codebase. This approach not only saved development time but also ensured that the system could be easily adapted to accommodate future changes without unnecessary complications. Overall, this design methodology, focused on modularity and optimization, provided a flexible, scalable system that could be adjusted based on changing requirements, making it ideal for future enhancements and expansions.

**Conclusion**

The journey to create this prototype was enriching and educational with a lot of opportunities to be opened toward the exciting potentials of IoT for creating affordable, customizable, and efficient security solutions. The integration of the ESP32-CAM, Arduino Uno, PIR sensor, and servo motor managed to realize an active, motion-responsive surveillance system that brings theory to life in this project. Overcoming the practical challenges in development showed that modular design, iterative testing, and the precise selection of components were important, each playing its role in refining the functionality of the system. The process also highlighted how academic research and online resources can provide valuable insights into component capabilities, integration strategies, and alternative approaches. That being said, the prototype promises immense possibilities for enhancement in the future: object detection using AI, motion-activated photo capturing and storing those images on SD cards with the help of the in-built SD card slot of ESP32 (on an inbuilt slot of ESP32, one can add a 4GB one) or with integration to cloud services such as Google Drive to safely back it up. This would also be in line with the diverse user needs and would improve the overall performance of the system. Further, these enhancements could be made cost-effectively compared to commercial solutions available in local markets. Moreover, the project opens up possibilities for integration with other smart home systems or remote monitoring applications. This may open the door to a more connected, automated surveillance experience. The study hereby recurrently elaborates on the viability of building a capable, scalable, and flexible security system using the widely available IoT components, therefore opening avenues for further innovations in smart surveillance and showcasing the transformative potential of accessible technology.

**Cost Break Down**

ESP32-CAM + OV2640 camera: Rs 650

Arduino Uno + Cable: Rs 900

PIR Sensor: Rs 100

Servo Motor: Rs 150

Optional device is ESP32-CAM-MB over FTDI because of ease of use.

ESP32-CAM-MB: Rs 500

Jumper Wires: Rs 50

Total: Rs 2350