

AI-102: Microsoft Certified Azure AI Engineer Associate

Smart CCTV Using Azure IoT Hub and Computer Vision

By

Miriam El Hage Sleiman

A Portfolio Project Created to Showcase Competencies and Skills Acquired from Microsoft's
AI Engineer Associate and Data Scientist Associate Certifications.

SUMMARY DESCRIPTION OF THE PROJECT

Project Developer:

Miriam El Hage Sleiman

Senior Mechatronics Engineering Student at AUST Lebanon

Project Title:

Smart CCTV Using Azure IoT Hub and Computer Vision

Azure Subscription:

Free trial

Date:

October 2025

Keywords:

IoT Integration, Computer Vision, Azure Cloud, Smart Surveillance

Abstract:

This personal project demonstrates the development and integration of a Smart CCTV system combining edge computing through Arduino with cloud-based AI services on Microsoft Azure. The setup uses a PIR motion sensor connected to an Arduino Uno for real-time motion detection, which triggers a Python client to capture video input and communicate with Azure IoT Hub. Motion events are logged and relayed to Azure, where Computer Vision services can be applied for further image analysis and recognition. This implementation also highlights the use of IoT Hub for device-to-cloud data exchange showcasing how Azure's AI services can enhance surveillance systems by enabling intelligent automation and visual analytics. This project acts as a showcase of the practical application of AI and machine learning concepts in intelligent monitoring scenarios as well as being a practical demonstration of skills aligned with Microsoft's AI Engineer Associate and Data Scientist Associate certifications.

TABLE OF CONTENTS

I.	Introduction.....	5
II.	Objective.....	5
III.	System Overview.....	5
IV.	Hardware Setup.....	6
V.	Cloud and Model Integration.....	7
VI.	Results and Discussion.....	8
VII.	Conclusion.....	10
VIII.	Evidence and Files.....	11
	References.....	12
	Appendix.....	13

LIST OF FIGURES

Figure 1. Diagram and simulation of the hardware on TinkerCAD.....	6
Figure 2. Hardware flowchart.....	7
Figure 3. The used resources.....	8
Figures 4,5. IoT Hub device and Computer Vision endpoints.....	9
Figure 6. CLI and JSON results for IoT Hub and Computer Vision services.....	10
Figure 7. IoT Hub in Azure Portal.....	11

I. Introduction

This project presents a prototype smart CCTV system that integrates an Arduino-based PIR motion sensor with Azure IoT Hub [1] and Computer Vision services. The system is a demonstration of a simplified edge-to-cloud workflow in which motion events detected at the hardware level trigger cloud-side image processing and object recognition. The implementation highlights how IoT devices and AI services can be combined to create intelligent surveillance applications.

II. Objective

The objective is to design and implement a compact smart surveillance prototype capable of detecting motion using a PIR sensor connected to an Arduino Uno, sending motion data to the cloud via Azure IoT Hub, capturing and analyzing images through a Python script linked to Azure Computer Vision as well as demonstrating event-driven AI inference from edge-generated data. Thus, practicing concepts learned from the AI-102 (Azure AI Engineer Associate) certification and applying them in an IoT-based scenario [2].

III. System Overview

The system consists of three main layers:

1. Edge Layer (Arduino + PIR Sensor): Detects motion and transmits an event signal.,
2. Cloud Communication Layer (Azure IoT Hub): Provides a secure channel for data exchange between the IoT device and the cloud service.,
3. Application Layer (Python + Azure Computer Vision): Receives event notifications, captures images from a webcam, and sends them to Computer Vision for object detection and labeling.

Upon motion detection, the Arduino sends a signal through IoT Hub. The Python client subscribed to the IoT Hub endpoint triggers the camera, captures a frame, and submits it to the Computer Vision API for analysis [3]. The detected objects or scene descriptions are then displayed and stored locally for reference.

IV. Hardware Setup

The hardware component consists of an Arduino Uno board and a PIR motion sensor module. The PIR sensor is responsible for detecting infrared radiation changes in its field of view, thus indicating movement. Connections: VCC → 5V on the Arduino, GND → GND on the Arduino, OUT → Digital Pin 2. To check for proper operation, the Arduino's onboard LED (Pin 13) was used as indicator. The Arduino continuously monitors the sensor output and sends a signal through the serial interface whenever motion is registered.

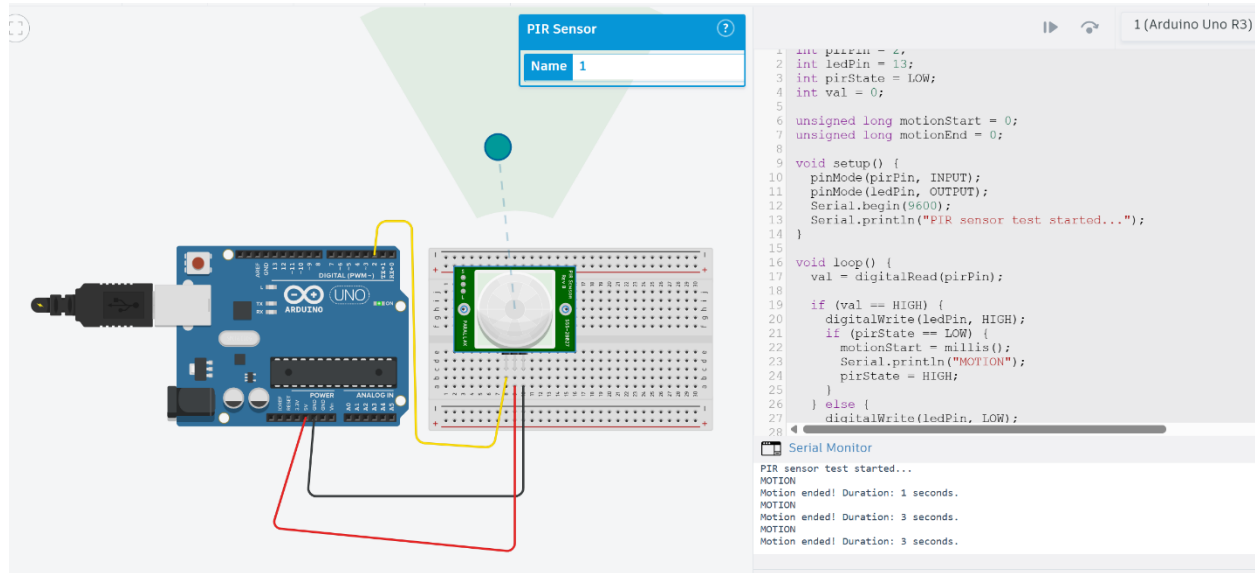


Figure 1. Diagram and simulation of the hardware on TinkerCAD.

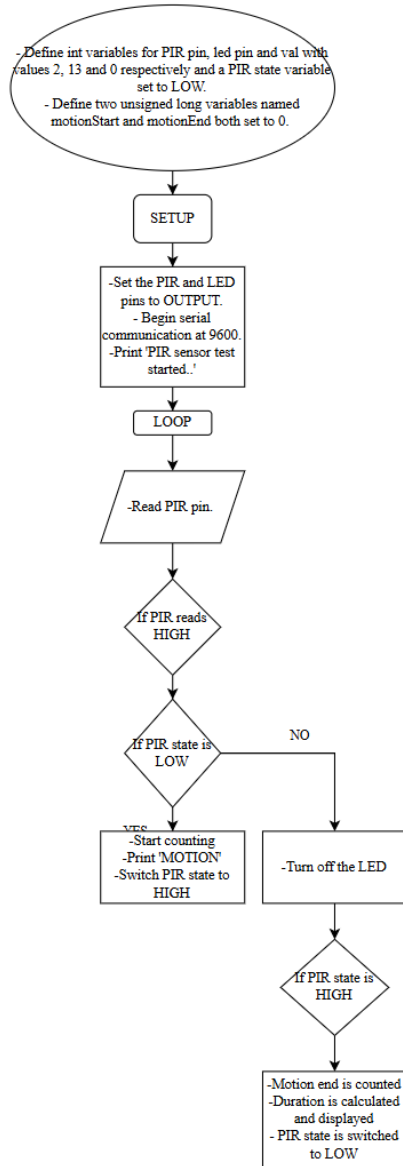


Figure 2. Hardware flowchart.

V. Cloud and Model Integration

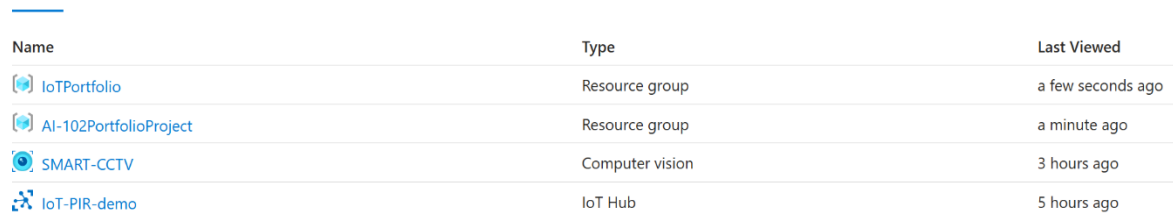
The cloud side of the system leverages two Azure services:

1. Azure IoT Hub:

The Arduino's motion signal is transmitted via serial link to a Python-based IoT client that publishes the event to an IoT Hub device endpoint. This is the bridge between the local hardware and Azure's AI services.

2. Azure Computer Vision (Cognitive Services):

Upon receiving a motion event, the Python script activates the computer's webcam to capture an image frame which is sent to the Computer Vision API and returns a JSON response containing detected objects, scene descriptions, as well as confidence levels. The results are logged and optionally saved as labeled snapshots for later review. This configuration demonstrates a seamless integration of event-driven IoT communication with cloud-based AI inference, illustrating the potential of Azure for intelligent edge solutions

A screenshot of the Azure portal showing a table of resources. The table has three columns: Name, Type, and Last Viewed. There are four rows of resources listed.





Name	Type	Last Viewed
 IoTPortfolio	Resource group	a few seconds ago
 AI-102PortfolioProject	Resource group	a minute ago
 SMART-CCTV	Computer vision	3 hours ago
 IoT-PIR-demo	IoT Hub	5 hours ago

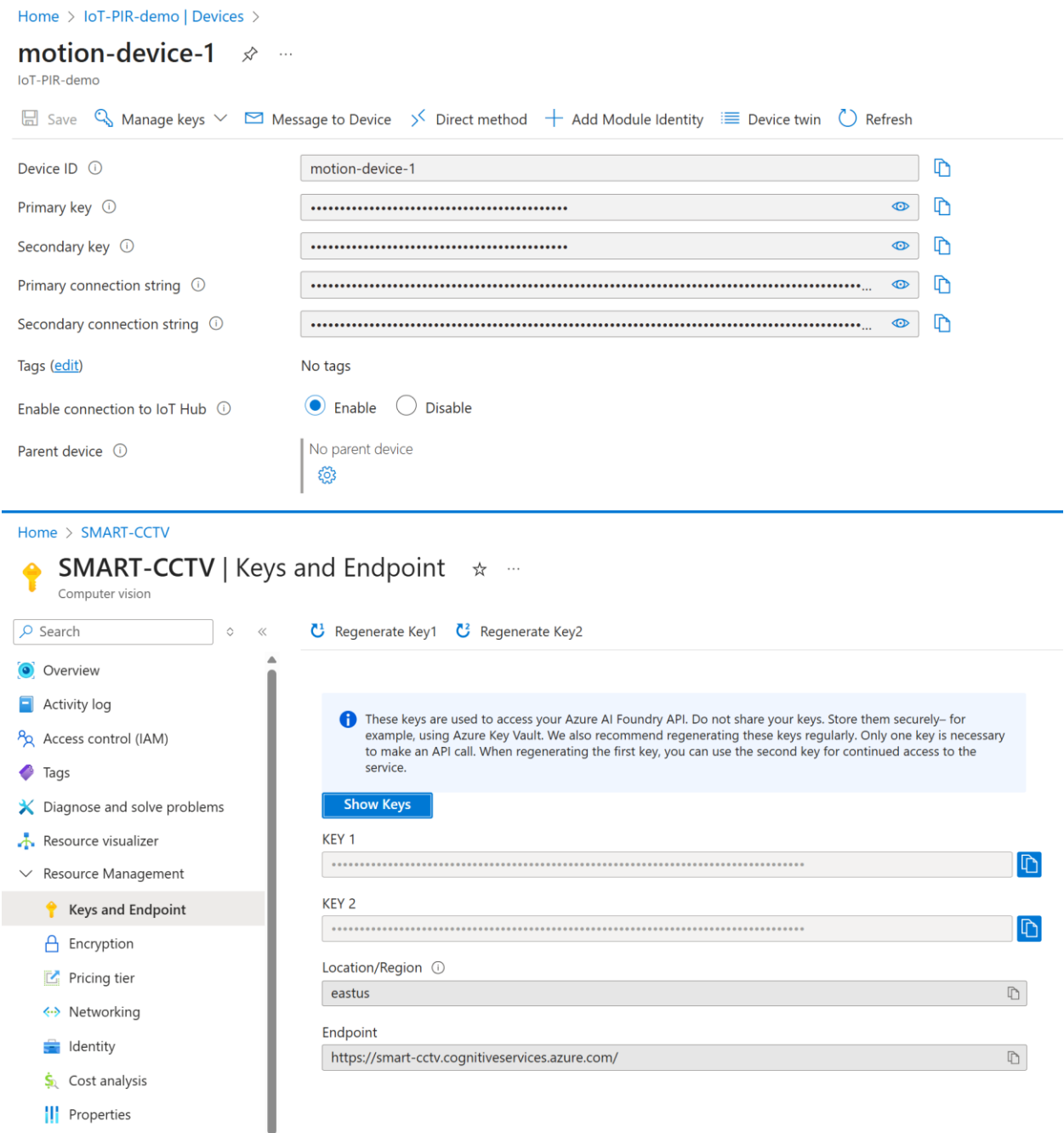
Figure 3. The used resources.

VI. Results and Discussion

The experiment successfully validated both the Arduino and Python components of the Smart CCTV prototype. On the hardware side, the Arduino Uno board operated as expected. In fact, the LED blinked consistently whenever I waved my hand in front of it, confirming that the microcontroller executed the code correctly and that the serial communication between the Arduino and computer was stable since the Python program would immediately execute by opening the computer's webcam.

On the software side, the Python client application was able to communicate effectively with Azure IoT Hub and Computer Vision service. Upon running the client, data transmissions from the Arduino were successfully captured and sent to the IoT Hub and Computer Vision endpoints. VSCode's terminal displayed a JSON result describing what Computer Vision had detected. On the other hand, an IoT Hub monitoring command was executed on Azure CLI to verify incoming messages. The console output confirmed that message calls were being received, indicating successful system integration between the local Python environment and the cloud service.

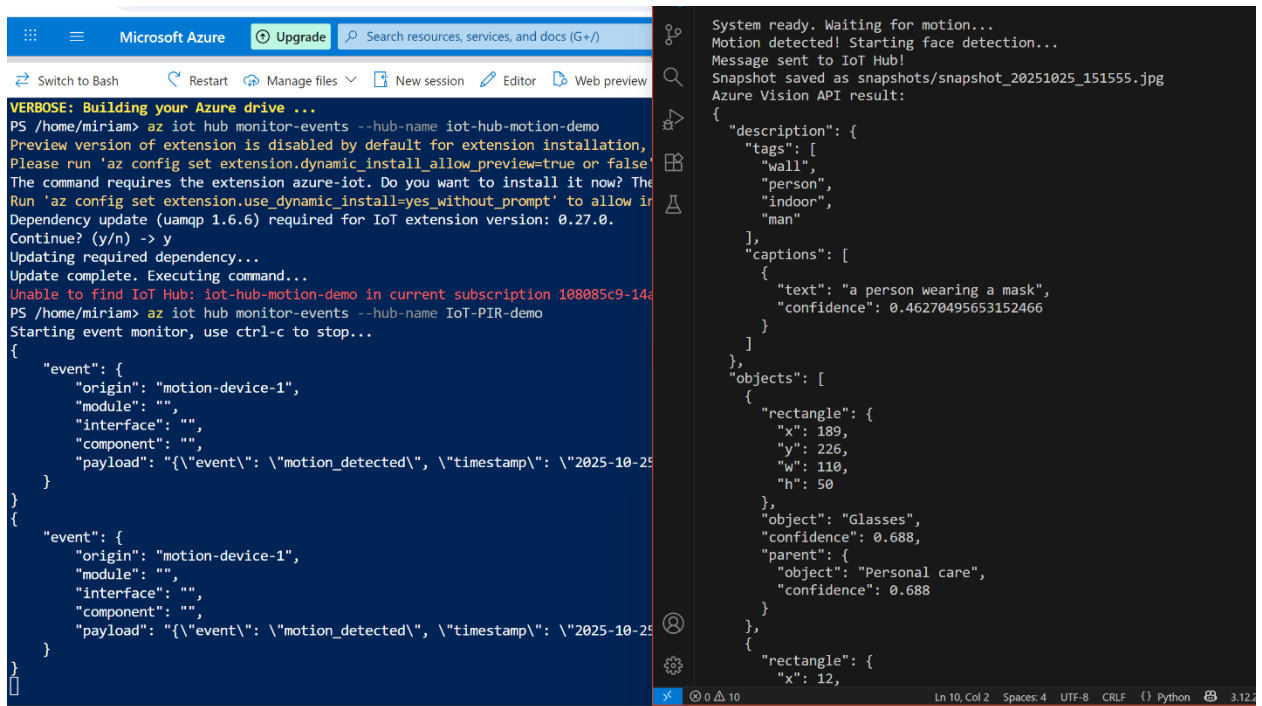
These results demonstrate that my Smart CCTV system achieved its core objective of establishing real-time communication between an edge device (Arduino) and a cloud-based IoT service (Azure IoT Hub). Both the local processing and cloud linkage operated reliably, forming the foundation for further expansion into computer vision analysis and automated event detection.



Figures 4,5. IoT Hub device and Computer Vision endpoints.

VII. Conclusion

This project successfully demonstrated a functional prototype of a Smart CCTV system that integrates edge and cloud computing using Arduino and Microsoft Azure services. The Arduino Uno, connected to a PIR motion sensor, was programmed to detect motion and send real-time data to a Python client application. The client then transmitted these updates to the Azure IoT Hub [4], enabling continuous monitoring and data logging from the local device to the cloud. The system's performance validated the feasibility of using low-cost hardware and accessible cloud tools to build intelligent surveillance systems. With the IoT Hub and Computer Vision services, the framework can be further extended to analyze camera feeds for object or face detection, creating an end-to-end edge-to-cloud AI application [5]. A detailed video demonstration of this project will accompany this report, showcasing the workflow and the functioning of both the Arduino and cloud integration. This documentation serves to supplement the video by outlining the technical process, implementation steps, and results obtained throughout the experiment.



```
Microsoft Azure Upgrade Search resources, services, and docs (G+/)
Switch to Bash Restart Manage files New session Editor Web preview

VERBOSE: Building your Azure drive ...
PS /home/miriam> az iot hub monitor-events --hub-name iot-hub-motion-demo
Preview version of extension is disabled by default for extension installation,
Please run 'az config set extension.dynamic_install_allow_preview=true or false'
The command requires the extension azure-iot. Do you want to install it now? The
Run 'az config set extension.use_dynamic_install=yes_without_prompt' to allow it
Dependency update (uamqp 1.6.6) required for IoT extension version: 0.27.0.
Continue? (y/n) -> y
Updating required dependency...
Update complete. Executing command...
Unable to find IoT Hub: iot-hub-motion-demo in current subscription 1088085c9-14a
PS /home/miriam> az iot hub monitor-events --hub-name IoT-PIR-demo
Starting event monitor, use ctrl-c to stop...
{
  "event": {
    "origin": "motion-device-1",
    "module": "",
    "interface": "",
    "component": "",
    "payload": "{\"event\": \"motion_detected\", \"timestamp\": \"2025-10-25\"}"
  }
}
{
  "event": {
    "origin": "motion-device-1",
    "module": "",
    "interface": "",
    "component": "",
    "payload": "{\"event\": \"motion_detected\", \"timestamp\": \"2025-10-25\"}"
  }
}

System ready. Waiting for motion...
Motion detected! Starting face detection...
Message sent to IoT Hub!
Snapshot saved as snapshots/snapshot_20251025_151555.jpg
Azure Vision API result:
{
  "description": {
    "tags": [
      "wall",
      "person",
      "indoor",
      "man"
    ],
    "captions": [
      {
        "text": "a person wearing a mask",
        "confidence": 0.46270495653152466
      }
    ]
  },
  "objects": [
    {
      "rectangle": {
        "x": 189,
        "y": 226,
        "w": 110,
        "h": 50
      },
      "object": "Glasses",
      "confidence": 0.688,
      "parent": {
        "object": "Personal care",
        "confidence": 0.688
      }
    },
    {
      "rectangle": {
        "x": 12,

```

Figure 6. CLI and JSON results for IoT Hub and Computer Vision services.

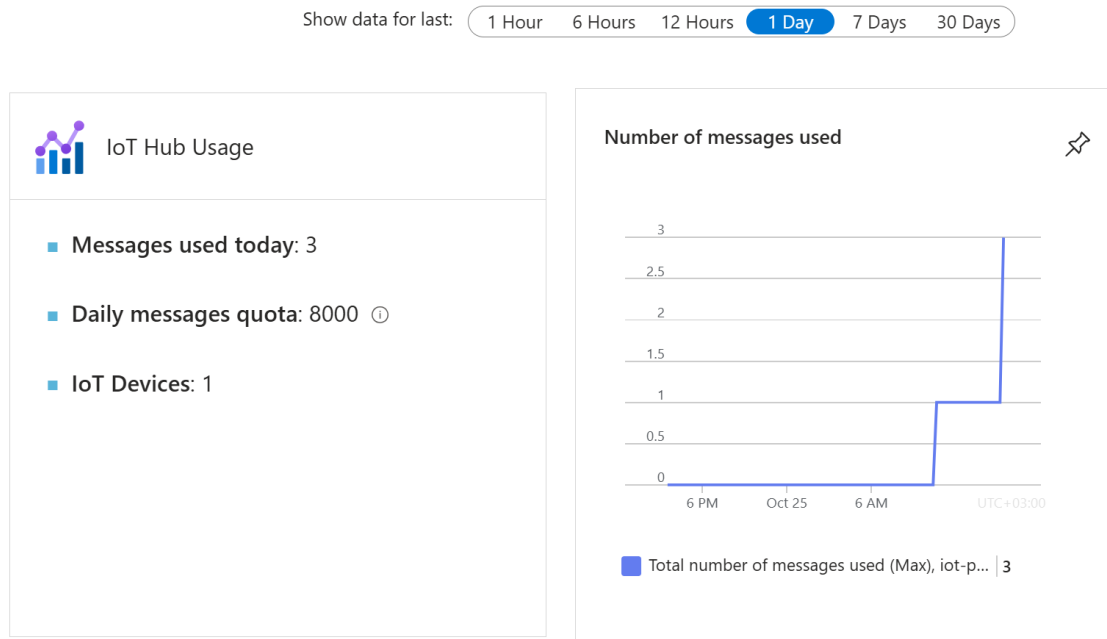


Figure 7. IoT Hub in Azure Portal.

VIII. Evidence and Files

Supporting evidence includes:

- Arduino and Python codes in the Appendix
- Diagram and flowchart.
- Screenshots.

References

- [1] Microsoft Learn, Azure IoT Hub Documentation. [Online]. Available: <https://learn.microsoft.com/en-us/azure/iot-hub/> [Accessed: Sept. 9, 2025]
- [2] Microsoft Learn, Microsoft Certified: Azure AI Engineer Associate – Certifications. [Online]. Available: <https://learn.microsoft.com/en-us/credentials/certifications/azure-ai-engineer/> [Accessed: Aug. 23, 2025]
- [3] Microsoft Learn, Azure AI Vision Documentation – Quickstarts, Tutorials, API Reference. [Online]. Available: <https://learn.microsoft.com/en-us/azure/ai-services/computer-vision/> [Accessed: Oct. 1, 2025]
- [4] Microsoft Learn, Azure IoT Hub Device and Service SDKs. [Online]. Available: <https://learn.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-sdks> [Accessed: Oct. 8, 2025]
- [5] Microsoft Learn, Designing and Implementing a Microsoft Azure AI Solution (AI-102T00). [Online]. Available: <https://learn.microsoft.com/en-us/training/courses/ai-102t00> [Accessed: Aug. 24, 2025]

Appendix

Python code (The endpoints and keys can no longer be used):

```
import os
from datetime import datetime
import cv2
import serial
import time
import requests
import json
from azure.iot.device import IoTHubDeviceClient, Message

AZURE_ENDPOINT = "https://smart-cctv.cognitiveservices.azure.com/"
AZURE_KEY =
"3n6KBKchWq4rOzKnCLF3u2NwBK9gzQfxF7uk0hXstubrdGq0ETrpJQQJ99BJACYeBjFXJ
3w3AAAFACOG94Zb"
VISION_URL = AZURE_ENDPOINT +
"vision/v3.2/analyze?visualFeatures=Objects,Description"

connection_str = "HostName=IoT-PIR-demo.azure-
devices.net;DeviceId=motion-device-
1;SharedAccessKey=N5X10CsFoVbo4yNKyd5JF0YNjI7gbW070KhyK4HzFj4="
device_client =
IoTHubDeviceClient.create_from_connection_string(connection_str)
device_client.connect()

ser = serial.Serial('COM3', 9600)
os.makedirs("snapshots", exist_ok=True)
cam = cv2.VideoCapture(0)
face_cascade = cv2.CascadeClassifier(cv2.data.harcascades +
'haarcascade_frontalface_default.xml')
```

```

print("System ready. Waiting for motion...")

while True:
    if ser.in_waiting > 0:
        line = ser.readline().decode('utf-8').strip()
        if line == "MOTION":
            print("Motion detected! Starting face detection...")
            msg = Message({'event': "motion_detected", "timestamp":
'' + datetime.now().isoformat() + ''})
            device_client.send_message(msg)
            print("Message sent to IoT Hub!")
            start_time = time.time()
            snapshot_taken = False
            while time.time() - start_time < 10:
                ret, frame = cam.read()
                if not ret:
                    break
                gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
                faces = face_cascade.detectMultiScale(gray, 1.3, 5)
                for (x, y, w, h) in faces:
                    cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0,
0), 2)

                    if not snapshot_taken:
                        timestamp =
datetime.now().strftime("%Y%m%d_%H%M%S")
                        filename =
f"snapshots/snapshot_{timestamp}.jpg"
                        cv2.imwrite(filename, frame)
                        print(f"Snapshot saved as {filename}")

                        # Azure Vision API call
                        with open(filename, "rb") as image_data:
                            headers = {

```

```

        "Ocp-Apim-Subscription-Key":
AZURE_KEY,
        "Content-Type": "application/octet-
stream"
    }
    response = requests.post(VISION_URL,
headers=headers, data=image_data)
    result = response.json()
    print("Azure Vision API result:")
    print(json.dumps(result, indent=2))

    snapshot_taken = True
    cv2.imshow("Face Detection", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
    print("Face detection session ended.")
    cam.release()
    cv2.destroyAllWindows()

```

Arduino code:

```

int pirPin = 2;
int ledPin = 13;
int pirState = LOW;
int val = 0;

unsigned long motionStart = 0;
unsigned long motionEnd = 0;

void setup() {
    pinMode(pirPin, INPUT);
    pinMode(ledPin, OUTPUT);
    Serial.begin(9600);
    Serial.println("PIR sensor test started...");
}

void loop() {

```

```

val = digitalRead(pirPin);

if (val == HIGH) {
    digitalWrite(ledPin, HIGH);
    if (pirState == LOW) {
        motionStart = millis();
        Serial.println("MOTION");
        pirState = HIGH;
    }
} else {
    digitalWrite(ledPin, LOW);
    if (pirState == HIGH) {
        motionEnd = millis();
        unsigned long duration = (motionEnd - motionStart) / 1000;
        Serial.print("Motion ended! Duration: ");
        Serial.print(duration);
        Serial.println(" seconds.");
        pirState = LOW;
    }
}
}

```