



# Building an IoT Security Camera With Raspberry Pi

---

Ashish Anil Bhogate - 1225513172

Gargi Sadashiv Gaitonde - 1226479100

# What is the task/problem you are trying to solve?

## **Motivation:**

Students living in rental houses often face the problem of stolen packages. This is because packages are often left unattended on the doorstep, making them easy targets for thieves.

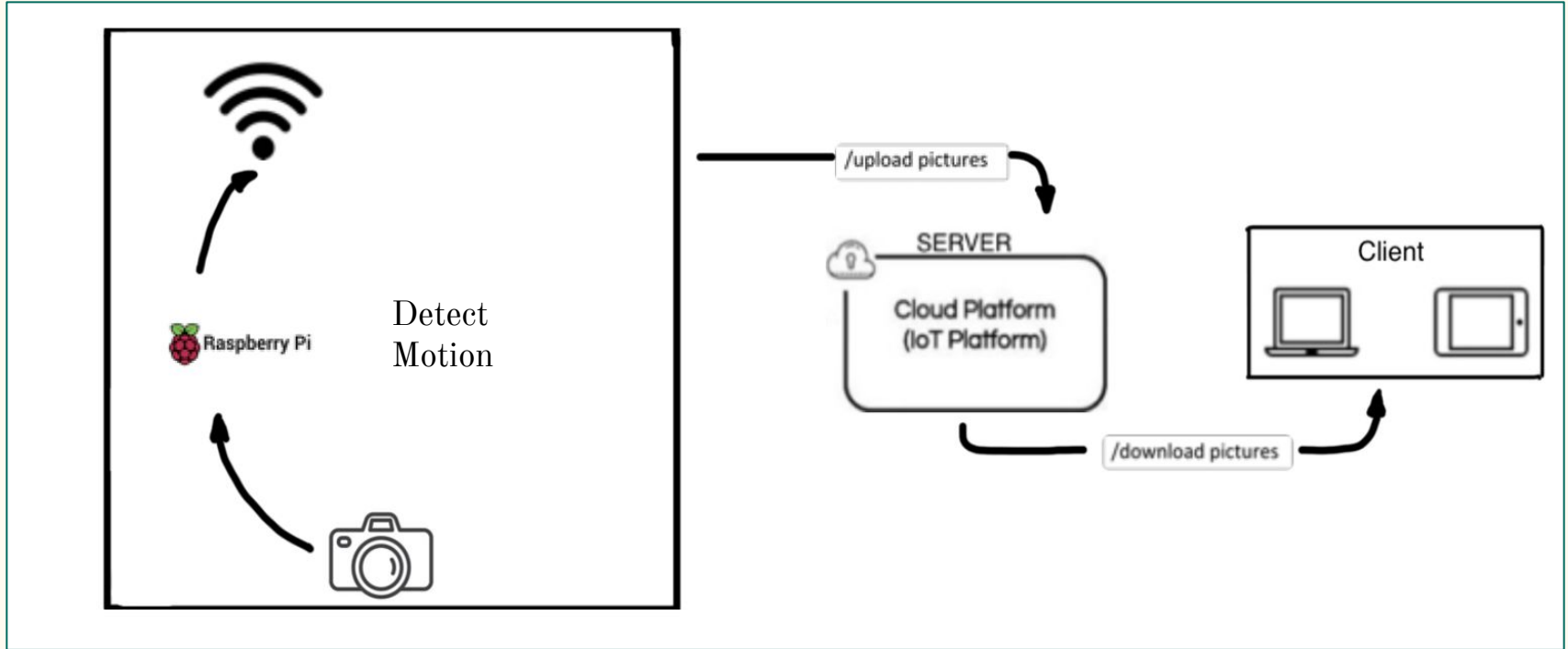
## **Current State of Smart Camera Control Systems**

- Issue 1: Not affordable
- Issue 2: Installation complexity

## **Project Objective**

Develop an open-source smart camera IOT system

# How do they interact with each other?



# Methodology

- Two-Fold Approach:
  - Development and testing on a laptop system with its camera module.
  - Standalone project executed on Raspberry Pi with its camera for versatile deployment.
- System Functionality:
  - Motion detection triggers image capture using a specialized module.
  - Captured images securely stored on a remote server.
  - Dedicated web page enables user access with timestamps for event review.
- Comparative Analysis:
  - Evaluation of system efficiency and performance between implementations.
  - Focus on time taken for motion detection, image capture, and secure storage.

# What are the tradeoffs you have made in your design?

**Performance:** The Raspberry Pi 3 is a powerful device, but it is not as powerful as a dedicated server. This means that the system may not be able to handle a large number of cameras.

**Security:** The system uses a number of security measures, but it is not completely secure. It is important to take steps to protect the system from unauthorized access.

**Accuracy:** The motion detection algorithm may not be able to classify unusual activity in all cases. This could lead to false positives or false negatives.

**Usability:** The system may be difficult to use for users who are not familiar with technology.

# Challenges Encountered

- Laptop (Local Device):
  - Meticulous adjustment of device drivers and OpenCV configurations for seamless webcam interfacing.
  - Ensured compatibility across various webcam models.
- Raspberry Pi:
  - Overcame compatibility issues with Pi Camera Module.
  - Optimized interaction and fine-tuned camera module settings for optimal performance.
- Sensitivity Optimization:
  - Conducted extensive experimentation to determine the optimal threshold for motion detection.
- Secure Communication with AWS EC2:
  - Key management for SSH connections and implemented SFTP for secure data transfer.
- Network Challenges Addressed:
  - Mitigated latency and packet loss issues through server and network settings configurations.

# Challenges Encountered

- Robust Exception Handling:
  - Implemented Python exception handling for motion detection and image upload scripts.
- Graceful Issue Handling:
  - Addressed network interruptions, server unavailability, and unexpected errors to prevent system failures and data loss.
- Server-Side Optimization:
  - Optimized file handling in the server-side script (server.js) for efficient storage and retrieval of uploaded images.
- Image Organization and Synchronization:
  - Ensured proper organization and synchronization of images on the server to prevent conflicts and streamline access.
- Calibration Procedures:
  - Fine-tuned motion detection parameters and camera settings for consistent performance across devices and environments.

# Implementation Discrepancies: Intended Plan vs. Actual Execution

## Intended Plan:

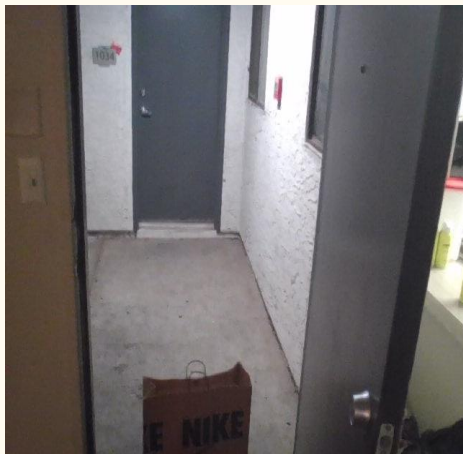
- Conduct motion detection on two systems: laptop with webcam and Raspberry Pi with Pi Camera Module.
- Compare algorithm efficiency between platforms.
- Assess image upload performance to AWS EC2.
- Record upload times for motion-detected images.

## Actual Execution:

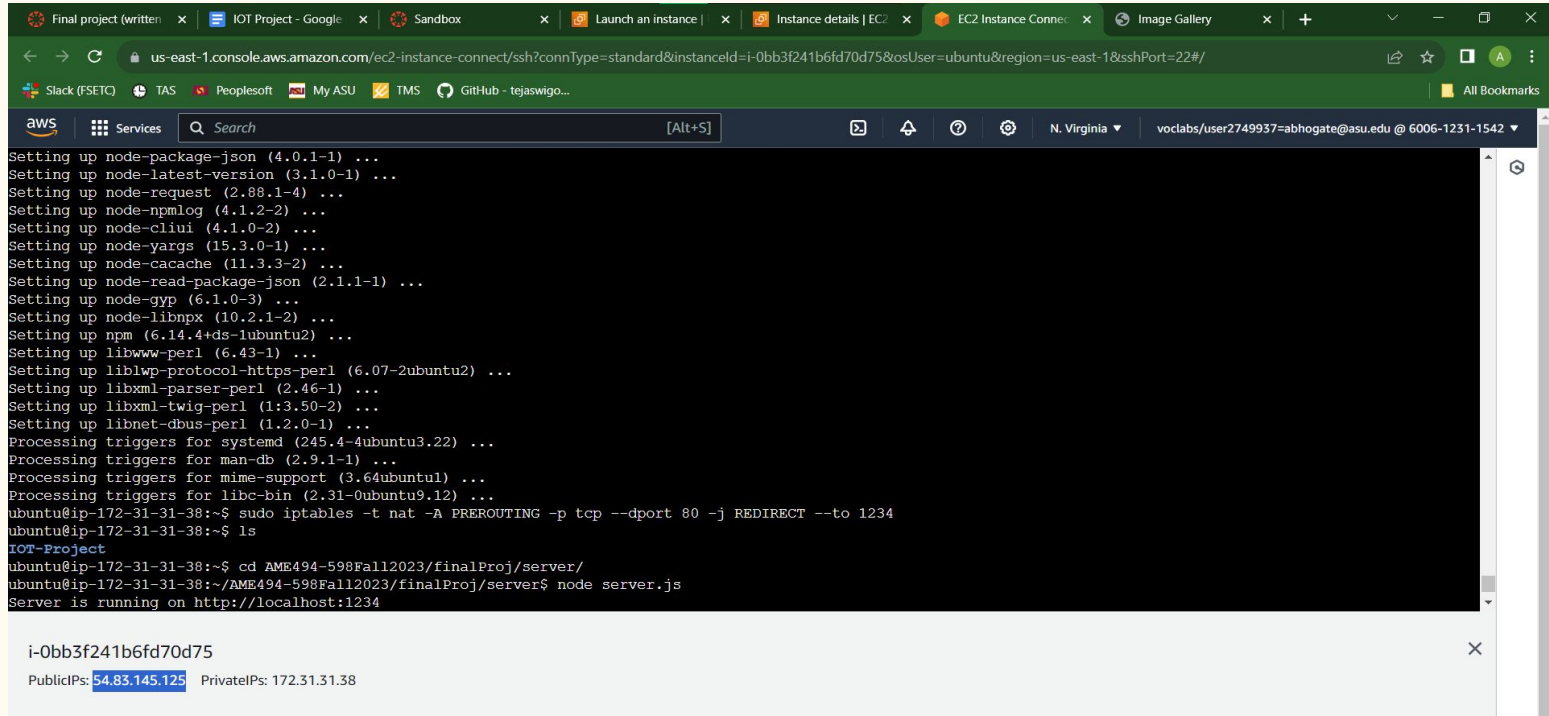
- Successful implementation addressing software and hardware challenges.
- Laptop: Motion detection via OpenCV with robust performance.
- Raspberry Pi: Seamless integration with Pi Camera Module, adapting to environmental changes.
- Recorded upload times on both systems for image transfers to AWS EC2.



# Motion Detected Results



# Starting AWS EC2 instance

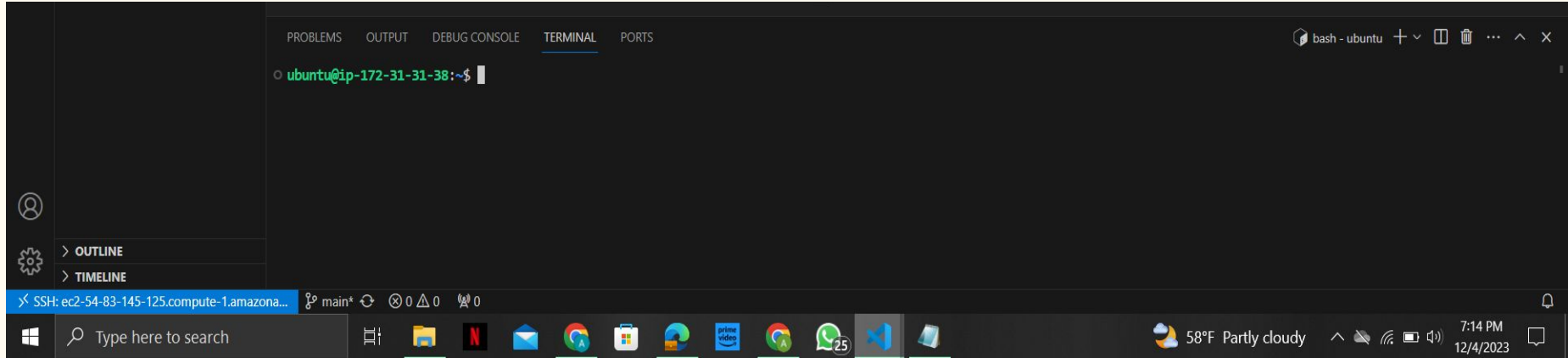


```
Setting up node-package-json (4.0.1-1) ...
Setting up node-latest-version (3.1.0-1) ...
Setting up node-request (2.88.1-4) ...
Setting up node-npmlog (4.1.2-2) ...
Setting up node-cliui (4.1.0-2) ...
Setting up node-yargs (15.3.0-1) ...
Setting up node-cacache (11.3.3-2) ...
Setting up node-read-package-json (2.1.1-1) ...
Setting up node-gyp (6.1.0-3) ...
Setting up node-libnpmx (10.2.1-2) ...
Setting up npm (6.14.4+ds-1ubuntu2) ...
Setting up libwww-perl (6.43-1) ...
Setting up liblwp-protocol-https-perl (6.07-2ubuntu2) ...
Setting up libxml-parser-perl (2.46-1) ...
Setting up libxml-twig-perl (1:3.50-2) ...
Setting up libnet-dbus-perl (1.2.0-1) ...
Processing triggers for systemd (245.4-4ubuntu3.22) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for mime-support (3.64ubuntu1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.12) ...
ubuntu@ip-172-31-31-38:~$ sudo iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to 1234
ubuntu@ip-172-31-31-38:~$ ls
IOT-Project
ubuntu@ip-172-31-31-38:~$ cd AME494-598Fall2023/finalProj/server/
ubuntu@ip-172-31-31-38:~/AME494-598Fall2023/finalProj/server$ node server.js
Server is running on http://localhost:1234
```

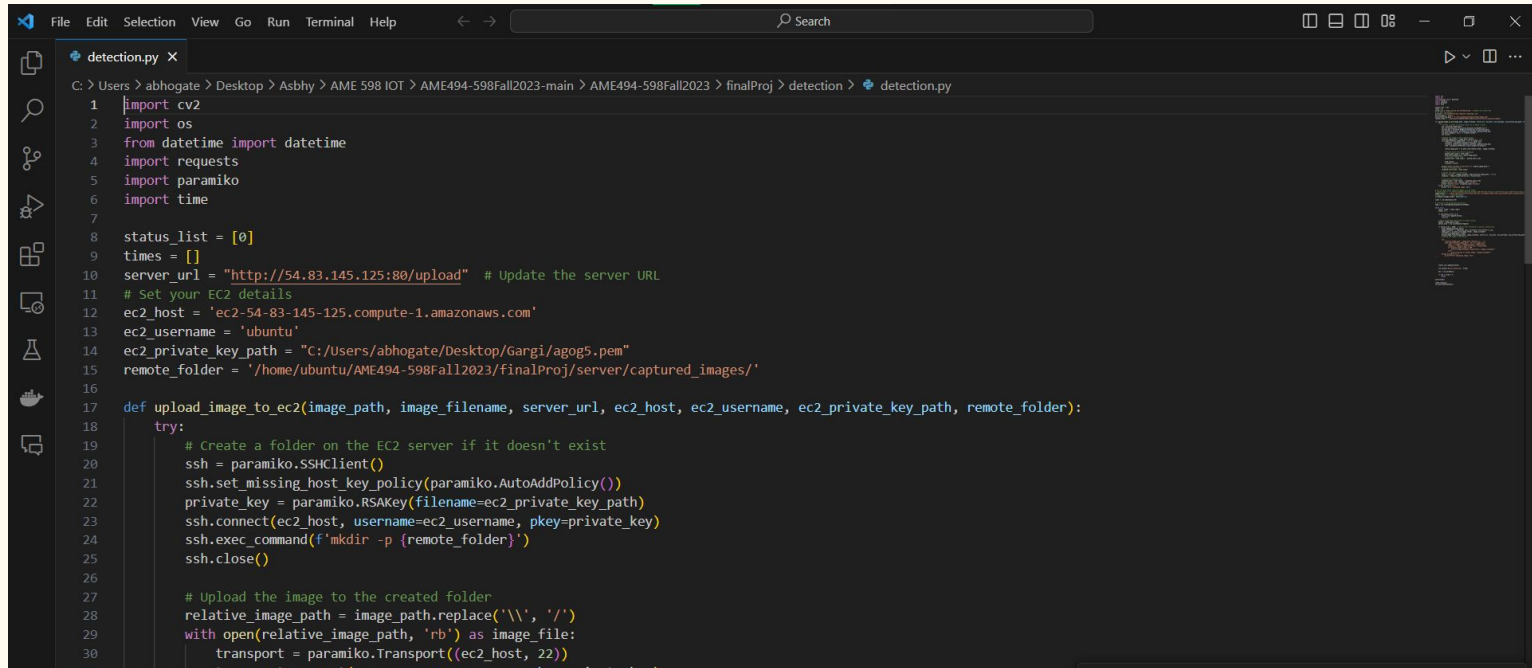
i-0bb3f241b6fd70d75

PublicIPs: 54.83.145.125 PrivateIPs: 172.31.31.38

# Connecting VS Code with EC2 instance using SSH



# Running Motion detection code, uploading images to the server and displaying server URL.



```
1 import cv2
2 import os
3 from datetime import datetime
4 import requests
5 import paramiko
6 import time
7
8 status_list = [0]
9 times = []
10 server_url = "http://54.83.145.125/upload" # Update the server URL
11 # Set your EC2 details
12 ec2_host = 'ec2-54-83-145-125.compute-1.amazonaws.com'
13 ec2_username = 'ubuntu'
14 ec2_private_key_path = "C:/Users/abhogate/Desktop/Gargi/agog5.pem"
15 remote_folder = '/home/ubuntu/AME494-598Fall2023/finalProj/server/captured_images/'
16
17 def upload_image_to_ec2(image_path, image_filename, server_url, ec2_host, ec2_username, ec2_private_key_path, remote_folder):
18     try:
19         # Create a folder on the EC2 server if it doesn't exist
20         ssh = paramiko.SSHClient()
21         ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
22         private_key = paramiko.RSAKey(filename=ec2_private_key_path)
23         ssh.connect(ec2_host, username=ec2_username, pkey=private_key)
24         ssh.exec_command(f'mkdir -p {remote_folder}')
25         ssh.close()
26
27         # Upload the image to the created folder
28         relative_image_path = image_path.replace('\\', '/')
29         with open(relative_image_path, 'rb') as image_file:
30             transport = paramiko.Transport((ec2_host, 22))
```

# Recording upload time and response time

```
PS C:\Users\abhogate\Desktop\Asbhy\AME 598 IOT\AME494-598Fall2023-main\AME494-598Fall2023\finalProj> & C:/Users/abhogate/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/abhogate/Desktop/Asbhy/AME 598 IOT/AME494-598Fall2023-main/AME494-598Fall2023/finalProj/detection/detection.py"
Image uploaded successfully to /home/ubuntu/AME494-598Fall2023/finalProj/server/captured_images/20231204190841.jpg
Upload time: 0.5416724681854248 seconds
Response time: 0.35111474990844727 seconds
Image uploaded successfully to /home/ubuntu/AME494-598Fall2023/finalProj/server/captured_images/20231204190845.jpg
Upload time: 0.41295886039733887 seconds
Response time: 0.34766101837158203 seconds
Image uploaded successfully to /home/ubuntu/AME494-598Fall2023/finalProj/server/captured_images/20231204190848.jpg
Upload time: 0.4229426383972168 seconds
Response time: 0.46049928665161133 seconds
Image uploaded successfully to /home/ubuntu/AME494-598Fall2023/finalProj/server/captured_images/20231204190853.jpg
Upload time: 0.4564998149871826 seconds
Response time: 0.5901570320129395 seconds
Image uploaded successfully to /home/ubuntu/AME494-598Fall2023/finalProj/server/captured_images/20231204190857.jpg
Upload time: 0.43903231620788574 seconds
```

# Displaying captured images on server URL

Final project (written) x IOT Project - Google x Sandbox x Launch an instance | x Instance details | EC2 x EC2 Instance Conne... x Image Gallery x +

Not secure | 54.83.145.125

Slack (FSETC) TAS Peoplesoft My ASU TMS GitHub - tejaswigo...

All Bookmarks

## Image Gallery


- [20231204190841.jpg](#) - Uploaded at 12/4/2023, 7:08:44 PM
- [20231204190845.jpg](#) - Uploaded at 12/4/2023, 7:08:48 PM
- [20231204190848.jpg](#) - Uploaded at 12/4/2023, 7:08:52 PM
- [20231204190853.jpg](#) - Uploaded at 12/4/2023, 7:08:57 PM
- [20231204190857.jpg](#) - Uploaded at 12/4/2023, 7:09:01 PM
- [20231204190901.jpg](#) - Uploaded at 12/4/2023, 7:09:04 PM
- [20231204190905.jpg](#) - Uploaded at 12/4/2023, 7:09:08 PM
- [20231204190909.jpg](#) - Uploaded at 12/4/2023, 7:09:12 PM

20231204190901.jpg (640x480) x +

Not secure | 54.83.145.125/captured\_images/20231204190901.jpg

Slack (FSETC) TAS Peoplesoft My ASU TMS GitHub - tejaswigo...

All Bookmarks



The image shows a woman with long dark hair, wearing a black jacket over a red top and dark pants, walking towards the camera in a hallway. She is carrying a large brown cardboard shopping bag with the Nike logo and the word 'NIKE' printed on it. The hallway has light-colored walls and a dark door frame on the right. A small sign with the number '1034' is visible on the wall to the left.

Windows taskbar: Type here to search, 58°F Partly cloudy, 7:17 PM 12/4/2023

# Future Improvements

## **Motion Detection System Enhancements:**

- Security Measures:
  - Implementation of robust authentication mechanisms
  - User credentials
  - Token-based access
  - Prevents unauthorized access to stored images on AWS EC2 server.
- Camera Module Upgrade:
  - Use advanced camera modules(Better resolution & optical features.)
  - Enhances visual clarity of captured images.
  - Dynamic Threshold Adjustments

# Conclusion

- Project successfully implemented a motion detection system on both a laptop with a webcam and a Raspberry Pi with the Pi Camera Module.
- The system demonstrated adaptability to diverse hardware configurations, effectively addressing challenges and showcasing robust performance.
- The project provided valuable insights into system efficiency.
- The lessons learned in overcoming challenges and optimizing performance contribute to a solid foundation for future projects at the intersection of computer vision, IoT, and cloud computing.



THANK YOU