Secure Eye: IOT Security Surveillance

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.

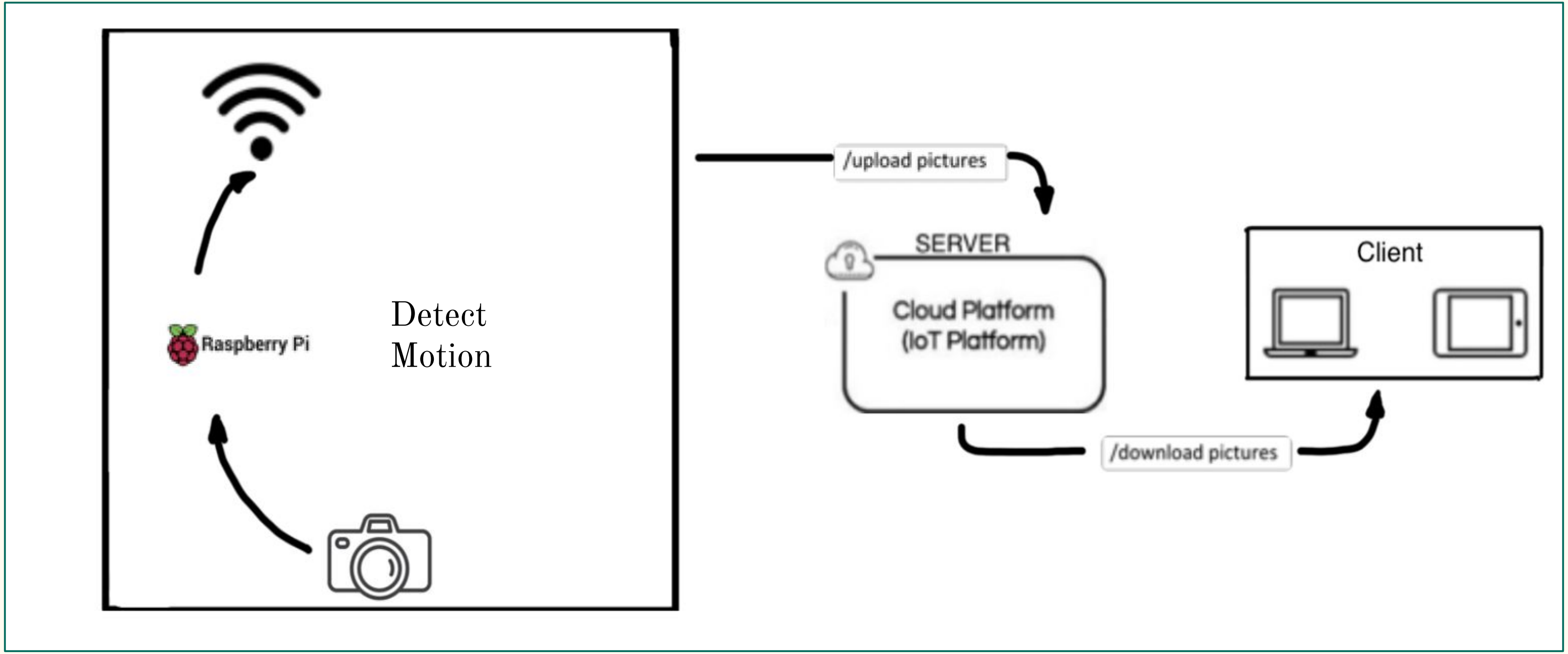
Current State of Smart Camera Control Systems

- 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 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:

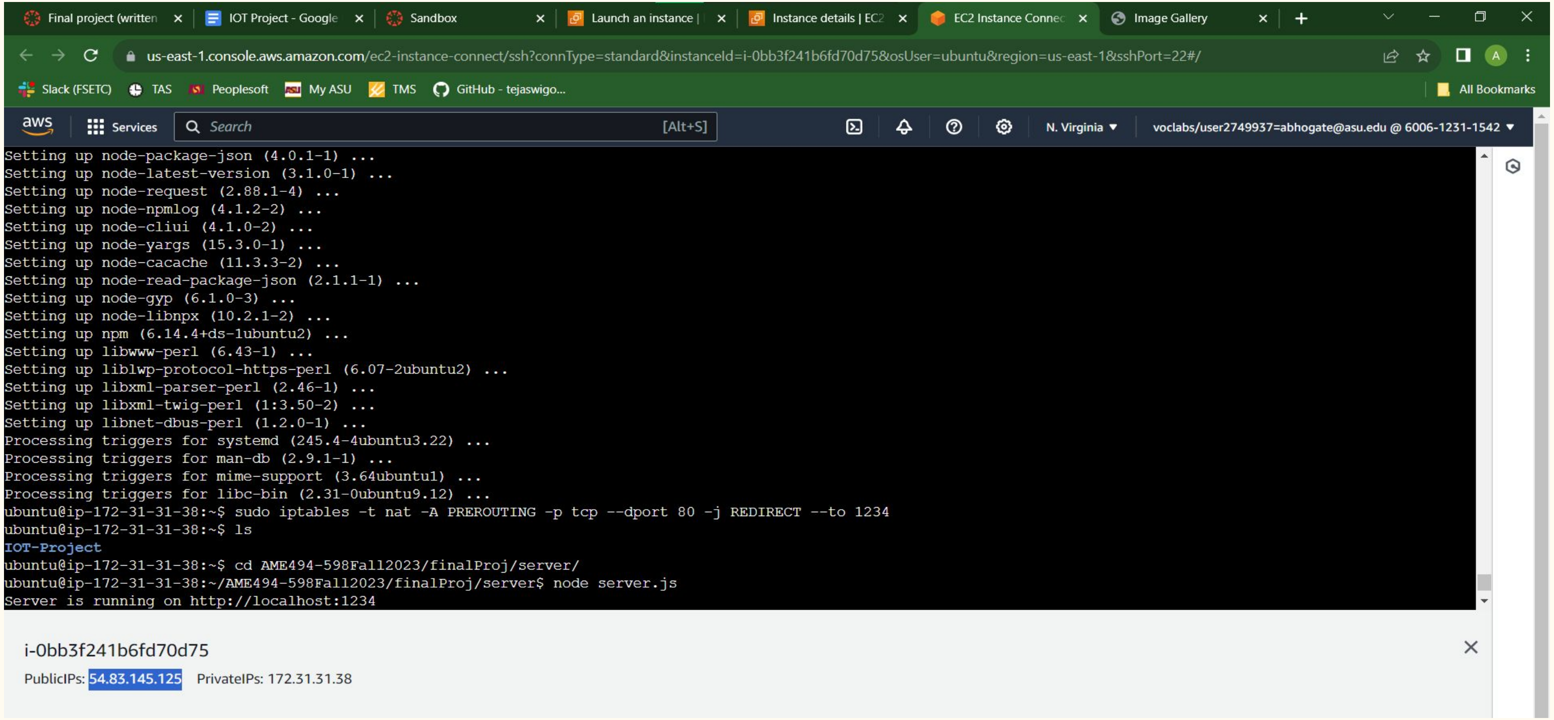
- Conduct motion detection on two systems: laptop with webcam and Raspberry Pi with Pi Camera Module.
- Compare algorithm efficiency between platforms.
- Assess image upload performance to AWS EC2.
- Record upload times for motion-detected images.

Actual Execution:

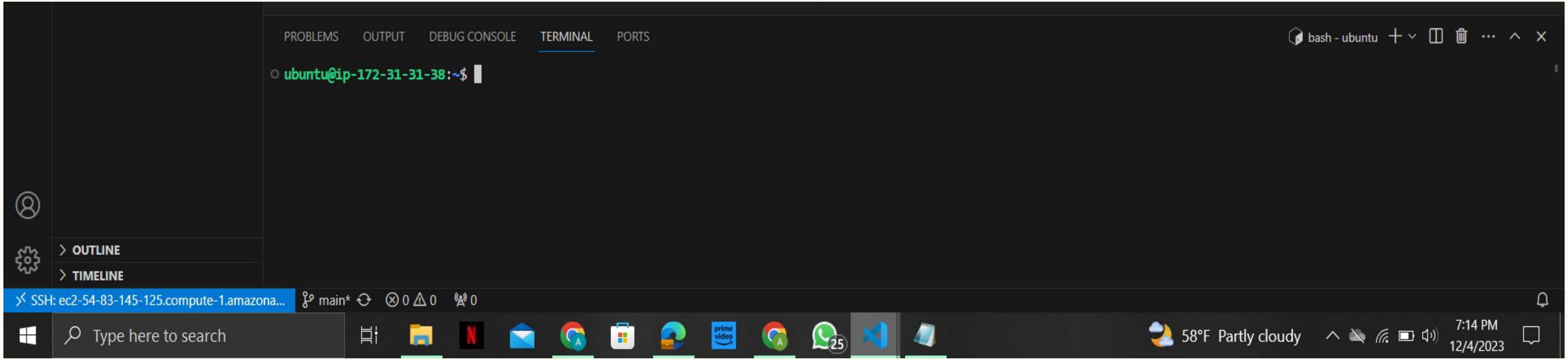
- Successful implementation addressing software and hardware challenges.
- Laptop: Motion detection via OpenCV with robust performance.
- Raspberry Pi: Seamless integration with Pi Camera Module, adapting to environmental changes.
- Recorded upload times on both systems for image transfers to AWS EC2.

Motion Detected Results

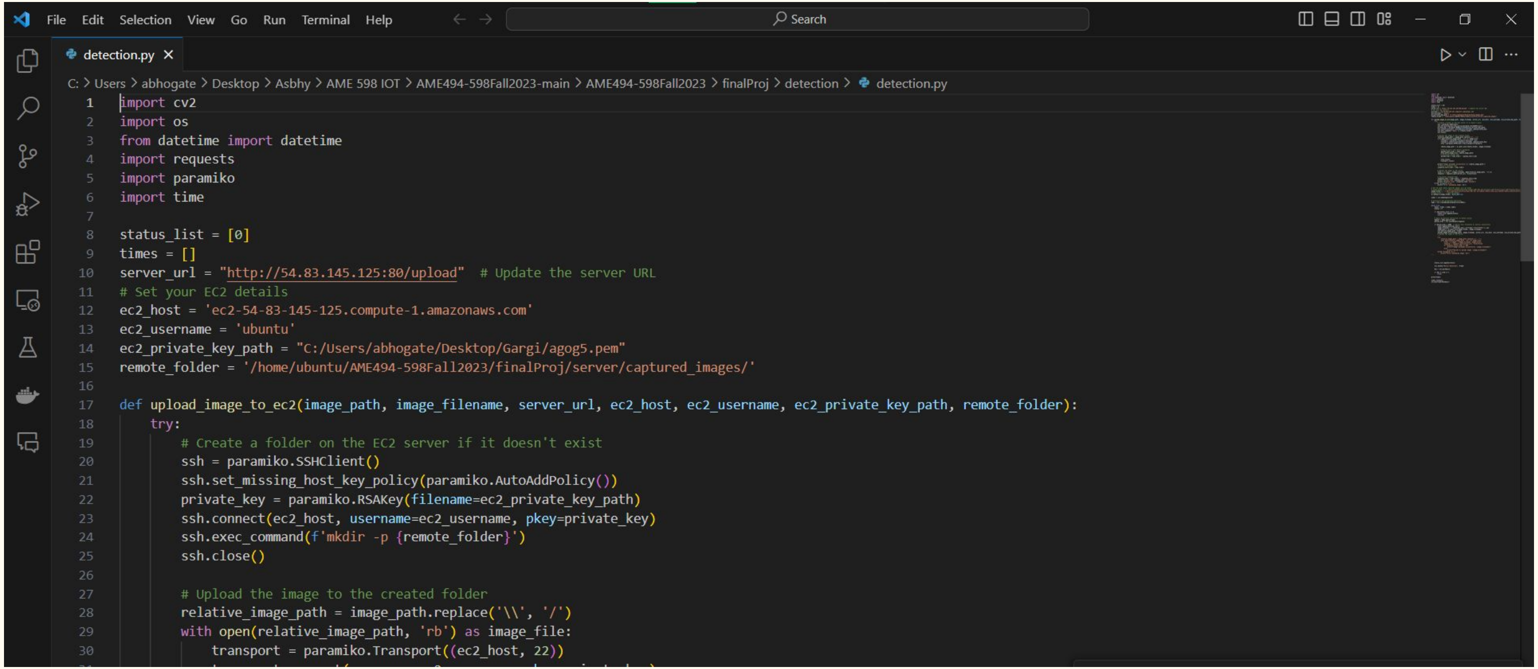
Starting AWS EC2 instance



Connecting VS Code with EC2 instance using SSH



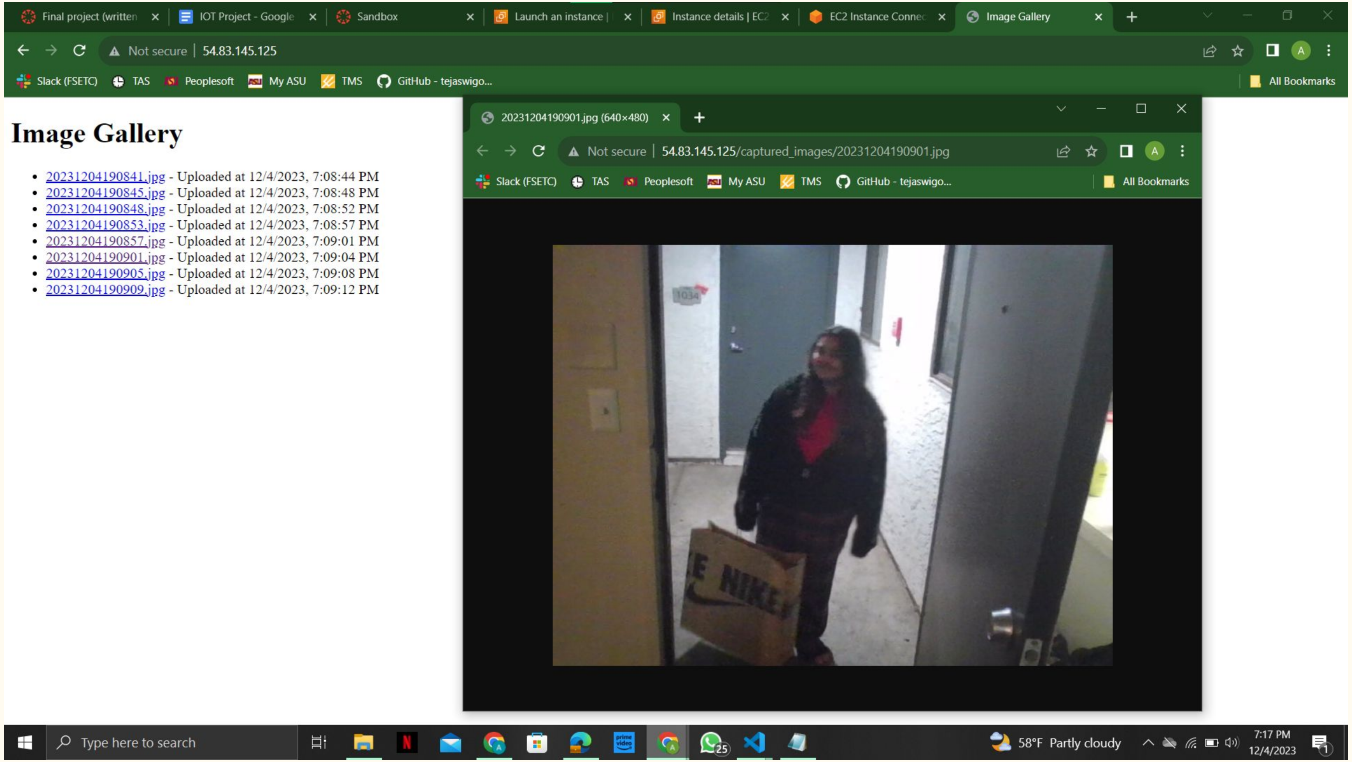
Running Motion detection code, uploading images to the server and displaying server URL.



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/Pyth
on311/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



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