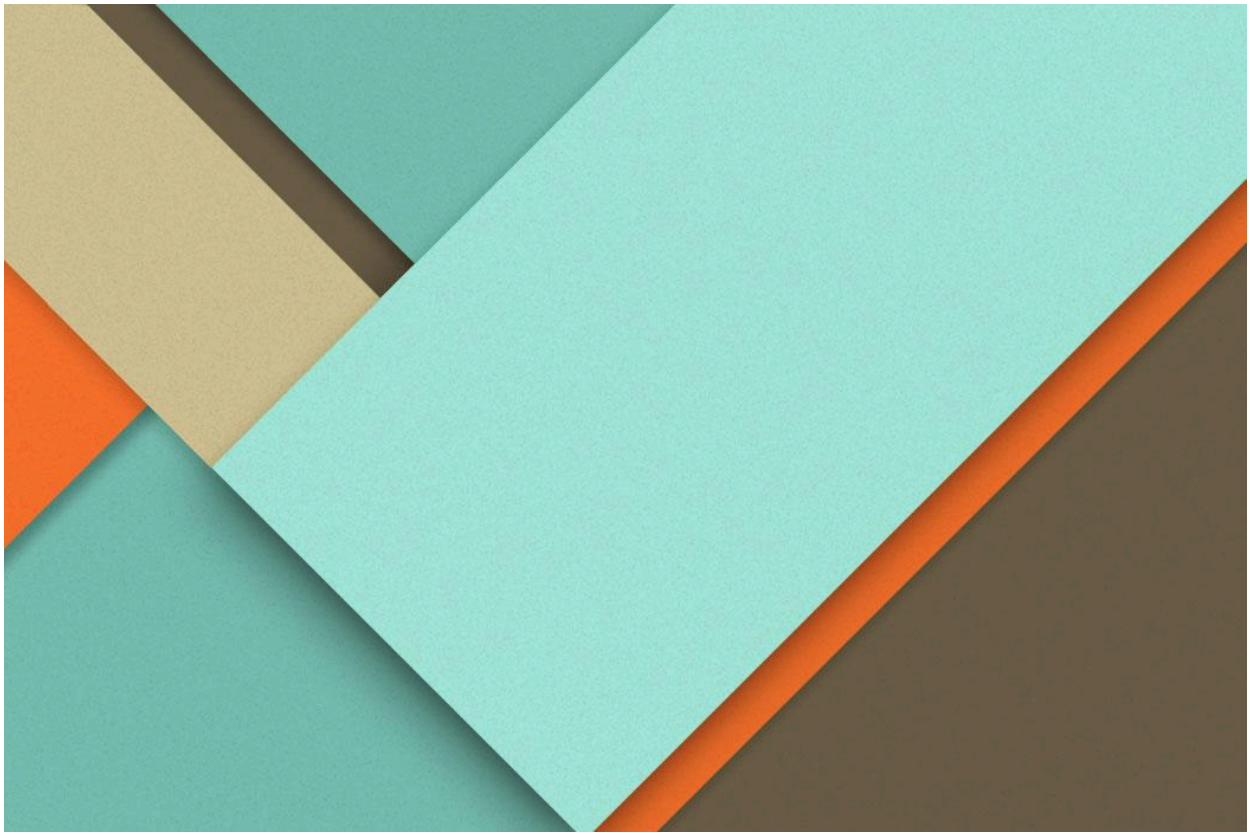

Future Enhancements for Pi Cam Setup

SIT782 - CapstoneTeam Project(B)

—

Ayush Kumar Som
222198016

Overview

This report outlines potential future enhancements for a Pi Cam setup, focusing on expanding the current local streaming and integration with a local host website to a globally accessible deployment. These enhancements will involve setting up global streaming, improving security, enhancing the user interface, and integrating additional features.

1. Global Streaming Deployment

Enhancement Details

Deploying the Pi Cam stream to a globally accessible website allows for remote monitoring from anywhere in the world. This requires setting up a server to host the video stream, configuring the Pi Cam for remote access, and ensuring the stream is secure.

Steps to Achieve Global Deployment

1. **Set Up a Global Server**
    - Choose a hosting provider (e.g., AWS, Google Cloud, Azure).
    - Set up a virtual machine (VM) or a containerized environment (e.g., Docker) to host the web server.
2. **Install and Configure Streaming Software**
    - Install web server software (e.g., Nginx, Apache) on the VM.
    - Install and configure streaming software (e.g., FFmpeg, GStreamer) to handle the video stream from the Pi Cam.
    - Configure the web server to serve the video stream. This might involve setting up an RTMP server using Nginx with the RTMP module.
3. **Configure Pi Cam for Remote Streaming**
    - Update the Pi Cam configuration to stream to the global server's RTMP endpoint.
    - Ensure the Pi Cam has a stable internet connection and configure any necessary port forwarding on the local network.
4. **Domain and SSL Certificate**
    - Register a domain name for your website.
    - Set up DNS to point to your VM's IP address.
    - Install an SSL certificate (e.g., Let's Encrypt) to secure the website.

2. Enhancing Security

Enhancement Details

Securing the video stream and the server is crucial to protect against unauthorized access and potential attacks.

Steps to Enhance Security

1. **Secure the Pi Cam and Network**
    - Change default credentials on the Pi Cam.
    - Use SSH for secure communication with the Pi Cam.
    - Set up a firewall on the Pi to restrict unnecessary traffic.
2. **Secure the Web Server**
    - Implement HTTPS using SSL/TLS certificates.
    - Use strong, unique passwords for server access.
    - Regularly update software to patch vulnerabilities.
3. **Stream Authentication**
    - Implement authentication for accessing the video stream.
    - Use tokens or username/password combinations to control access.

3. Improving User Interface

Enhancement Details

A user-friendly interface enhances the viewing experience and allows for better interaction with the live stream.

Steps to Improve the UI

1. **Responsive Web Design**
    - Use responsive web design techniques to ensure the website works well on different devices and screen sizes.
    - Utilize frameworks like Bootstrap or Materialize for consistency and ease of development.
2. **Interactive Features**
    - Add controls for pausing, playing, and rewinding the stream.
    - Implement a live chat feature for real-time communication.
3. **Analytics and Monitoring**
    - Integrate analytics tools (e.g., Google Analytics) to monitor website traffic and user behavior.
    - Display stream health metrics (e.g., latency, bitrate) for users.

4. Additional Features

Enhancement Details

Incorporating additional features can enhance the functionality and utility of the Pi Cam setup.

Steps to Add Additional Features

1. **Motion Detection**
    - Integrate motion detection software (e.g., OpenCV) to trigger alerts or recordings when motion is detected.
    - Configure the Pi Cam to send notifications (e.g., email, SMS) upon motion detection.
2. **Cloud Storage Integration**
    - Set up cloud storage (e.g., AWS S3, Google Drive) for recording and storing video footage.
    - Implement a system for managing and retrieving recorded footage.
3. **Multi-Camera Support**
    - Expand the system to support multiple Pi Cams streaming to the same server.
    - Create a dashboard for managing and viewing multiple streams simultaneously.

Detailed Steps for Each Enhancement

1. **Setting Up a Global Server**
    1. **Choose a Hosting Provider**
        - Example: AWS EC2, Google Cloud Compute Engine, Azure VM.
        - Create an account and set up a new instance.
    2. **Install Web Server Software**
        - SSH into the instance.
        - Install Nginx: `sudo apt update && sudo apt install nginx`
        - Install FFmpeg: `sudo apt install ffmpeg`
    3. **Configure Nginx for RTMP Streaming**
        - Edit Nginx configuration to include RTMP module.
        - Restart Nginx: `sudo systemctl restart nginx`

```
http {
    server {
        listen 80;
        server_name yourdomain.com;

        location / {
            root /var/www/html;
            index index.html;
        }
    }
}

rtmp {
    server {
        listen 1935;
        chunk_size 4096;

        application live {
            live on;
            record off;
        }
    }
}
```

    4. **Configure Pi Cam for Remote Streaming**
        - Install necessary packages on the Pi: `sudo apt install ffmpeg`

- Update the streaming script on the Pi to use the RTMP endpoint.

```
raspivid -o - -t 0 -w 1280 -h 720 -fps 25 | ffmpeg -re -ar 44100 -ac 2 -acodec pcm_s16le -f s16le -ac 2 -i - -vcodec h264 -pix_fmt yuv420p -g 50 -f flv rtmp://yourdomain.com/live/stream
```

2. **Enhancing Security**
    1. **Change Default Credentials**
        - Change the default password: `sudo passwd pi`
        - Disable password authentication and use SSH keys: `sudo nano /etc/ssh/sshd_config`
    2. **Set Up a Firewall**
        - Install ufw: `sudo apt install ufw`
        - Allow necessary ports: `sudo ufw allow 22`, `sudo ufw allow 80`, `sudo ufw allow 1935`
        - Enable the firewall: `sudo ufw enable`
    3. **Implement HTTPS**
        - Install Certbot: `sudo apt install certbot python3-certbot-nginx`
        - Obtain and install a certificate: `sudo certbot --nginx`
    4. **Stream Authentication**
        - Add basic authentication in Nginx.

```
location /live {
    auth_basic "Restricted Access";
    auth_basic_user_file /etc/nginx/.htpasswd;
}
```

- Create the password file: `sudo htpasswd -c /etc/nginx/.htpasswd user`

3. **Improving the User Interface**
    1. **Responsive Web Design**
        - Use HTML, CSS, and JavaScript frameworks like Bootstrap.

```
<html>
<head>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
</head>
<body>
    <div class="container">
        <h1>Live Stream</h1>
        <video id="live-stream" controls>
            <source src="rtmp://yourdomain.com/live/stream" type="video/mp4">
        </video>
    </div>
</body>
</html>
```

    2. **Interactive Features**
        - Use JavaScript libraries to add interactive elements.
        - Example: Integrate a chat feature using WebSockets.
    3. **Analytics Integration**

- Add Google Analytics tracking code to your website.

```
<script async src="https://www.googletagmanager.com/gtag/js?id=UA-XXXXXXXXX-X"></script>
<script>
    window.dataLayer = window.dataLayer || [];
    function gtag(){dataLayer.push(arguments);}
    gtag('js', new Date());
    gtag('config', 'UA-XXXXXXXXX-X');
</script>
```

4. **Adding Additional Features**

1. Motion Detection

- **Install OpenCV on the Pi:** sudo apt install python3-opencv
- **Create a Python script for motion detection and alerts:** (Image instructions not displayed)

```
import cv2

cap = cv2.VideoCapture(0)

while cap.isOpened():
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, (21, 21), 0)
    if motion_detected(blur):
        send_alert()
    cv2.imshow('Motion Detection', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

2. Cloud Storage Integration

- **Use AWS SDK to upload footage to S3:** (Image instructions not displayed)

```
import boto3

s3 = boto3.client('s3')
s3.upload_file('local_file.mp4', 'bucket_name', 'remote_file.mp4')
```

3. Multi-Camera Support

Enhancement Details

Expanding your Pi Cam setup to support multiple cameras involves configuring each Pi Cam to stream to a central server and updating the web interface to display multiple streams simultaneously. This can be useful for monitoring multiple areas or angles.

Steps to Achieve Multi-Camera Support

1. **Configure Each Pi Cam for Streaming:** Set up each Pi Cam with a unique stream key. (Image instructions not displayed)

```
# For Camera 1
raspivid -o - -t 0 -w 1280 -h 720 -fps 25 | ffmpeg -re -ar 44100 -ac 2 -acodec pcm_s16le -f s16le -ac 2 -i -
-vcodec h264 -pix_fmt yuv420p -g 50 -f flv rtmp://yourdomain.com/live/camera1

# For Camera 2
raspivid -o - -t 0 -w 1280 -h 720 -fps 25 | ffmpeg -re -ar 44100 -ac 2 -acodec pcm_s16le -f s16le -ac 2 -i -
-vcodec h264 -pix_fmt yuv420p -g 50 -f flv rtmp://yourdomain.com/live/camera2
```

2. **Update Nginx Configuration for Multiple Streams:** Modify the Nginx configuration to handle multiple RTMP streams. (Image instructions not displayed)

```
rtmp {
    server {
        listen 1935;
        chunk_size 4096;

        application live {
            live on;
            record off;

            # Camera 1
            stream camera1 {
                live on;
                record off;
            }

            # Camera 2
            stream camera2 {
                live on;
                record off;
            }
        }
    }
}
```

3. **Expand the Web Interface:** Update the web interface to display multiple video streams. Use HTML and JavaScript to create a layout that can handle multiple video elements. (Image instructions not displayed)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
    <title>Multi-Camera Stream</title>
    <style>
        .video-container {
            display: flex;
            flex-wrap: wrap;
            justify-content: space-around;
        }
        .video-item {
            margin: 10px;
            width: 45%;
        }
        video {
            width: 100%;
            height: auto;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1 class="text-center my-4">Multi-Camera Stream</h1>
        <div class="video-container">
            <div class="video-item">
                <h2>Camera 1</h2>
                <video id="camera1" controls autoplay>
                    <source src="rtmp://yourdomain.com/live/camera1" type="application/x-mpegURL">
                </video>
            </div>
            <div class="video-item">
                <h2>Camera 2</h2>
                <video id="camera2" controls autoplay>
                    <source src="rtmp://yourdomain.com/live/camera2" type="application/x-mpegURL">
                </video>
            </div>
        </div>
    </div>
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>
</body>
</html>
```

4. **Optimize Network and Server Resources:** Ensure your server and network can handle the increased load from multiple streams. Monitor resource usage and scale up your server resources if needed.
5. **Testing and Validation:** Test each camera stream individually and then together to ensure smooth performance. Validate that the web interface displays all streams