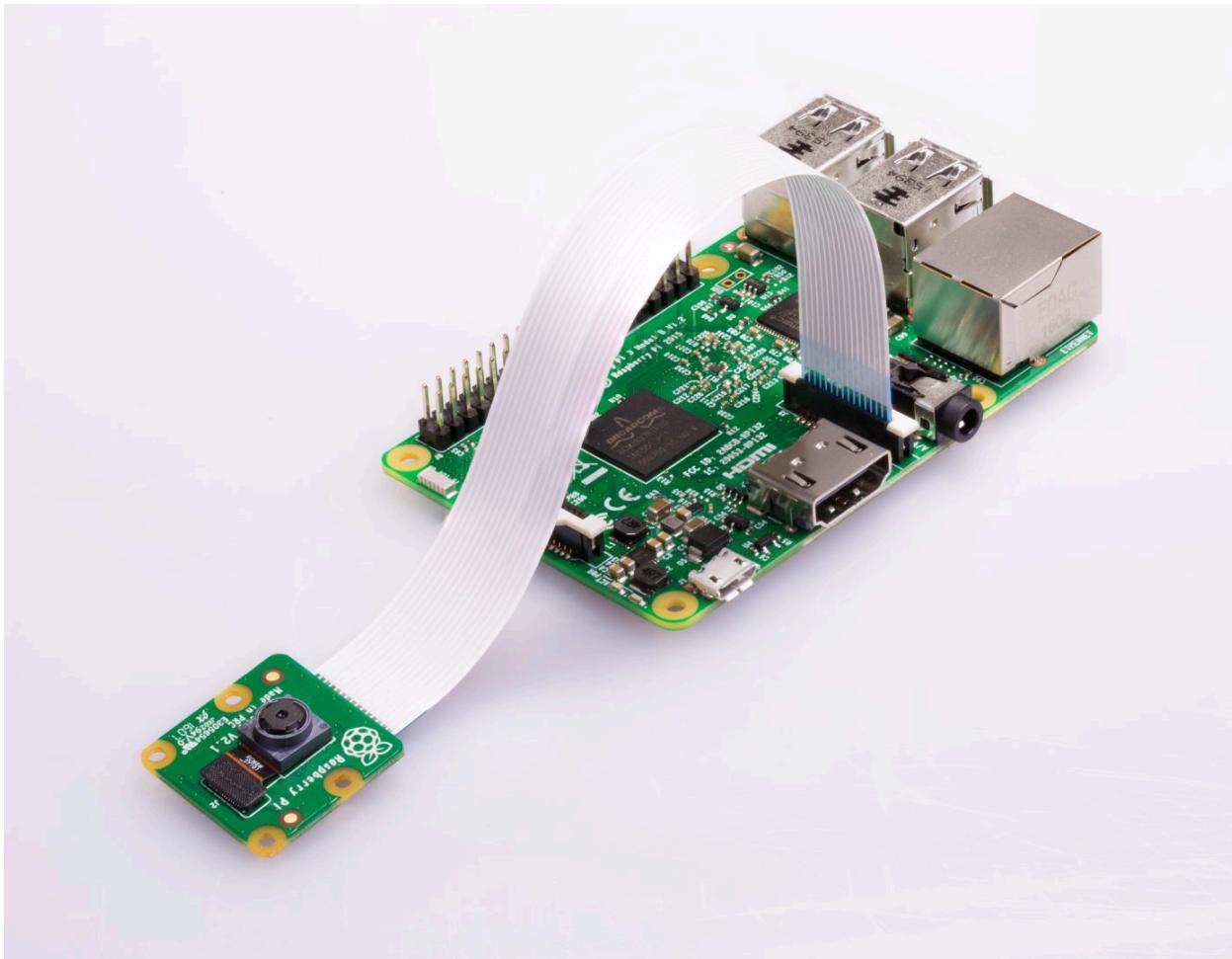


Pi Cam and Raspberry Pi setup

Ayush Kumar Som
22219801

1. Introduction

The Raspberry Pi, a versatile and affordable mini-computer, in combination with the Pi Camera (Pi Cam), provides a powerful platform for live-streaming video content. This guide explores the process of capturing live footage with a Pi Cam and streaming it on a website, making it accessible to a broad audience.



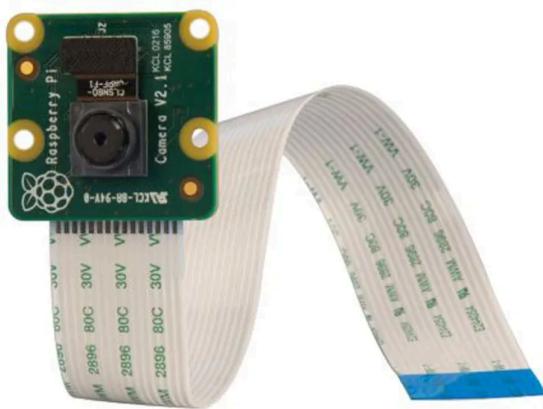
2. Hardware Setup

Components Required:

- Raspberry Pi 4



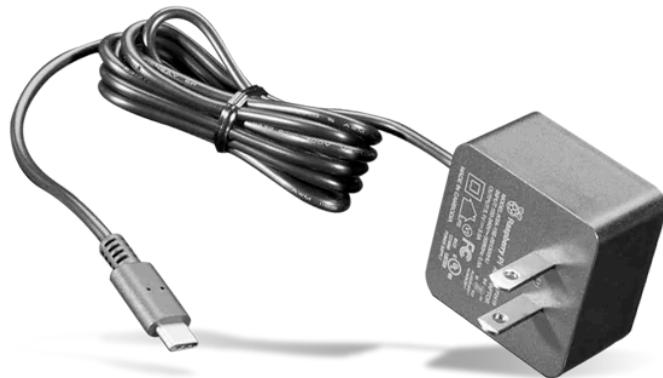
- Raspberry Pi Camera Module (Pi Cam)



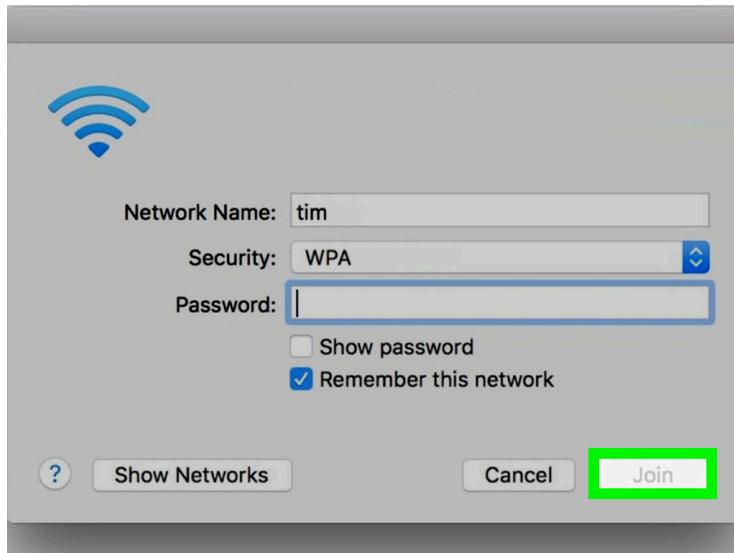
- MicroSD card (minimum 16GB, Class 10 recommended)



- 
- Power supply for Raspberry Pi



- Internet connection



Assembling the Hardware:

1. **Mount the Pi Cam:** Carefully attach the Pi Cam to the Raspberry Pi's camera port, ensuring the ribbon cable is securely connected.
 2. **Prepare the MicroSD Card:** Flash the Raspberry Pi OS onto the MicroSD card using software like Raspberry Pi Imager.
 3. **Power Up:** Insert the MicroSD card into the Raspberry Pi, connect the power supply, and boot up the device.
-

3. Software Setup

Configuring the Camera:

1. **Enable Camera Support:** Use 'sudo raspi-config' to enable the camera in the Interface Options.
2. **Test the Camera:** Capture a test image with 'raspistill -o test.jpg' to verify the camera is working.

Installing Streaming Software:

1. **Update and Upgrade:** Run 'sudo apt update' and 'sudo apt upgrade' to ensure all packages are up to date.
2. **Install FFmpeg:** Install FFmpeg by running 'sudo apt install ffmpeg', which is a versatile tool to handle video streaming.

4. Setting Up the Stream

Using FFmpeg for Live Streaming:

1. **Capture and Stream:** Use FFmpeg to capture video from the Pi Cam and stream it. A basic command to start streaming

Create a streaming Directory

```
mkdir ~/streaming  
cd ~/streaming
```

Start Streaming

```
'ffmpeg -f v4l2 -i /dev/video0 -f mpegts udp://YOUR_SERVER_IP:PORT'
```

2. **Adjust Parameters:** Experiment with different FFmpeg parameters to optimize the stream quality and performance based on your network conditions.

5. Steps to follow to Setup Streaming

After assembling your hardware as per the initial steps, you'll need to ensure your Raspberry Pi is ready for the task.

1. Enable Camera Support

1. Open Raspberry Pi Configuration Tool
`sudo raspi-config`
2. Navigate to Interface Options > Camera and enable it.
3. Reboot your Raspberry Pi to apply changes.

2. Installing FFmpeg

FFmpeg is a powerful tool that will handle the video capture and streaming.

`sudo apt update`

`sudo apt install ffmpeg`

3. Capturing Video from the Pi Cam and Streaming

This step involves using FFmpeg to capture video from the Pi Camera and stream it. For simplicity, we'll stream using the HLS (HTTP Live Streaming) protocol, which is widely supported on web platforms.

Stream with FFmpeg

1. Create a Streaming Directory
This directory will store the HLS segments and playlist.
`mkdir ~/streaming`
`cd ~/streaming`
2. Start Streaming
Replace YOUR_STREAM_KEY with your unique stream identifier. This command captures video from the camera and starts an HLS stream.
`ffmpeg -f v4l2 -i /dev/video0 -c:v libx264 -preset veryfast -maxrate 3000k -bufsize 6000k -pix_fmt yuv420p -g 50 -c:a aac -b:a 160k -ac 2 -ar 44100 -f hls -hls_time 2 -hls_playlist_type event stream.m3u8`

This command configures FFmpeg to:

- 
- Use the video4linux2 (v4l2) input device, which is standard for capturing video on Linux, including the Raspberry Pi.
 - Encode the video using the libx264 codec, which is efficient and widely supported.
 - Set a fast encoding preset to reduce latency.
 - Configure video bitrate and buffer sizes for quality and smooth streaming.
 - Output the stream in the HLS format, which is compatible with many web players.

4. Setting Up a Simple HTTP Server to Serve the Stream

You can use Python's HTTP server module to serve the stream directly from your Raspberry Pi. This is a straightforward method to make your stream accessible over the network.

```
cd ~/streaming
```

```
python3 -m http.server
```

This command starts a simple HTTP server on port 8000, serving files out of the `~/streaming` directory. You can access the stream by navigating to `http://<Raspberry_Pi_IP>:8000/stream.m3u8` from a device on the same network.

5. Embedding the Stream in a Web Page

To display your live stream on a webpage, you can use the HTML5 `<video>` tag. However, since we're using HLS, you might need a JavaScript library like hls.js to ensure compatibility across all browsers.

Sample HTML with hls.js

1. Add hls.js to Your Web Page

Include the hls.js library in your HTML header:

```
<script src="https://cdn.jsdelivr.net/npm/hls.js@latest"></script>
```

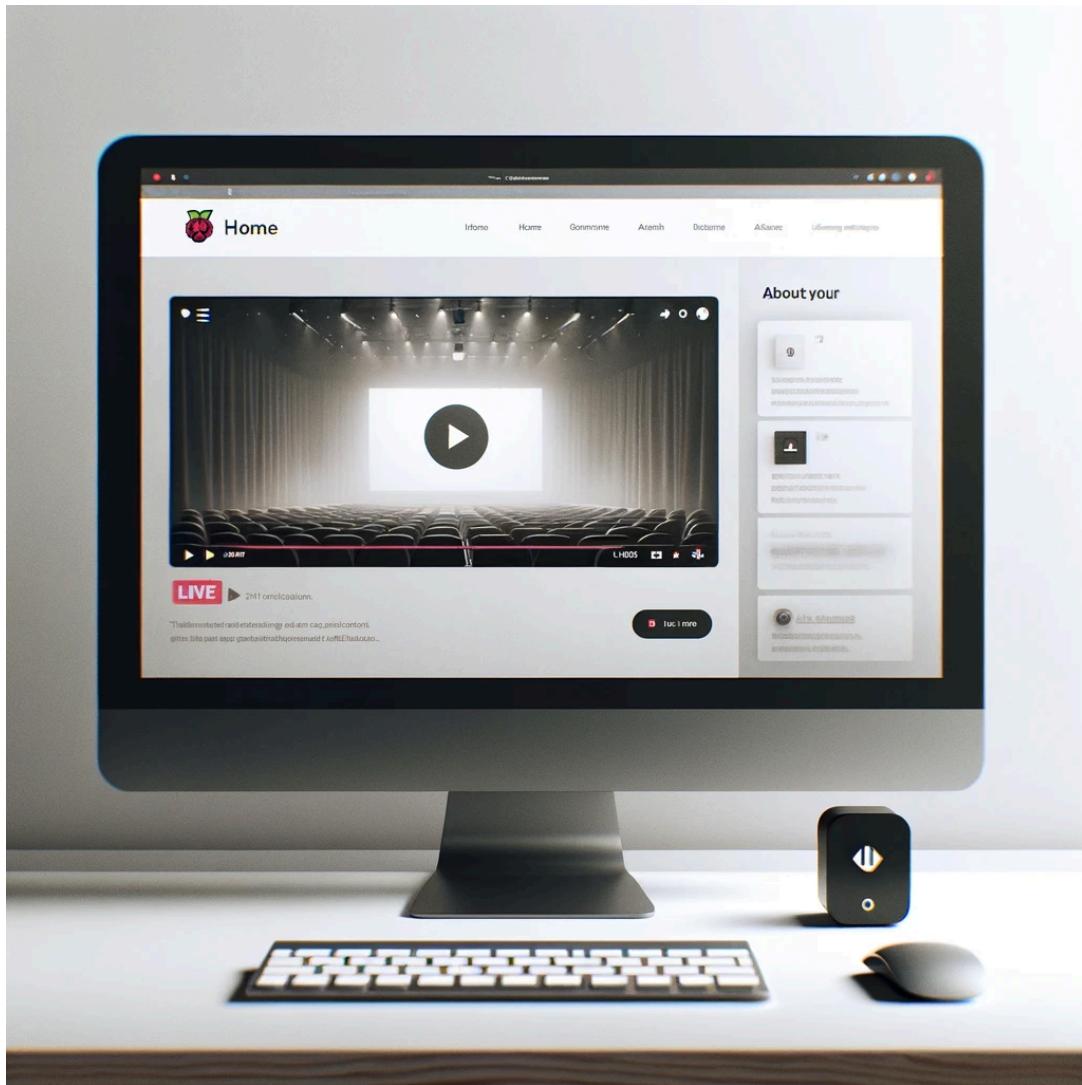
2. Embed Your Live Stream

Add a video element and script to initialize hls.js:

```
<video id="video" controls autoplay></video>
<script>
  var video = document.getElementById('video');
  var videoSrc = 'http://<Raspberry_Pi_IP>:8000/stream.m3u8';
  if (Hls.isSupported()) {
    var hls = new Hls();
```

```
hls.loadSource(videoSrc);
hls.attachMedia(video);
} else if (video.canPlayType('application/vnd.apple.mpegurl')) {
    video.src = videoSrc;
}
</script>
```

Replace <Raspberry_Pi_IP> with your Raspberry Pi's IP address. This script checks if the browser supports HLS natively or through hls.js and plays the stream accordingly.



6. Website Integration

Embedding the Live Stream:

1. **Streaming Server:** Set up a streaming server that can ingest the stream from the Raspberry Pi and serve it to web clients. NGINX with the RTMP module is a popular choice.
2. **Web Player:** Embed a video player on your website to play the live stream. Video.js is a popular open-source player that supports various streaming formats.

Sample HTML Code:

```
<video id="liveStream" controls>
    <source src="http://YOUR_SERVER_IP:PORT/live/stream.m3u8" type="video/mp4">
</video>
```

Security Measures:

- Implement HTTPS for your website to secure the viewer's connection.
- Use authentication and access control for the streaming server to restrict unauthorised access.

7. Troubleshooting and Optimization

- **Network Bandwidth:** Ensure your network bandwidth is sufficient for live streaming, considering both upload from the Raspberry Pi and download by viewers.
- **Latency:** Use low-latency streaming protocols like WebRTC if real-time interaction is crucial.
- **Power Supply:** Ensure a stable power supply for the Raspberry Pi, especially for long-duration streams.

Conclusion

This guide provides a foundation for setting up a live-streaming project with a Raspberry Pi and Pi Cam. With this setup, you can embark on various projects, from home surveillance systems to wildlife monitoring or live event broadcasting. Experiment with different configurations and software to tailor the system to your needs and optimize the streaming quality.