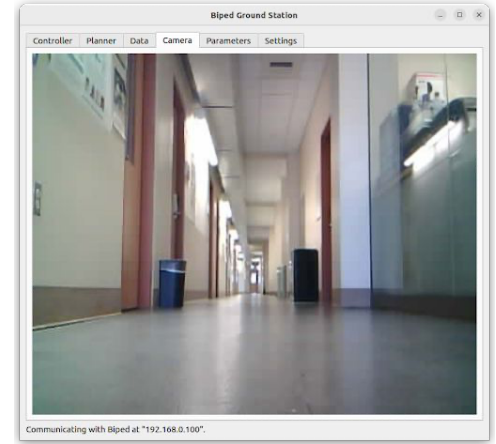
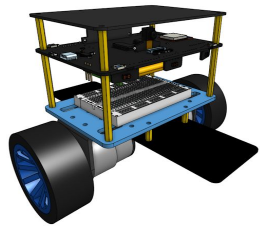


Approach - System Design

- **Biped**
 - Two-wheeled self-balancing robot running on ESP32 MCU
 - Biped will compress and send the images to Biped Ground Station
- **Biped Ground Station**
 - Remote host application for communicating with Biped
 - Receives and decompresses images from Biped
- **Lossy Network**
 - We can set the transmission power used by our router
 - We can vary the distance between Biped and the router



Biped Ground Station



Biped



Problem Description

We aim to compare and contrast images compression techniques for IoT devices:

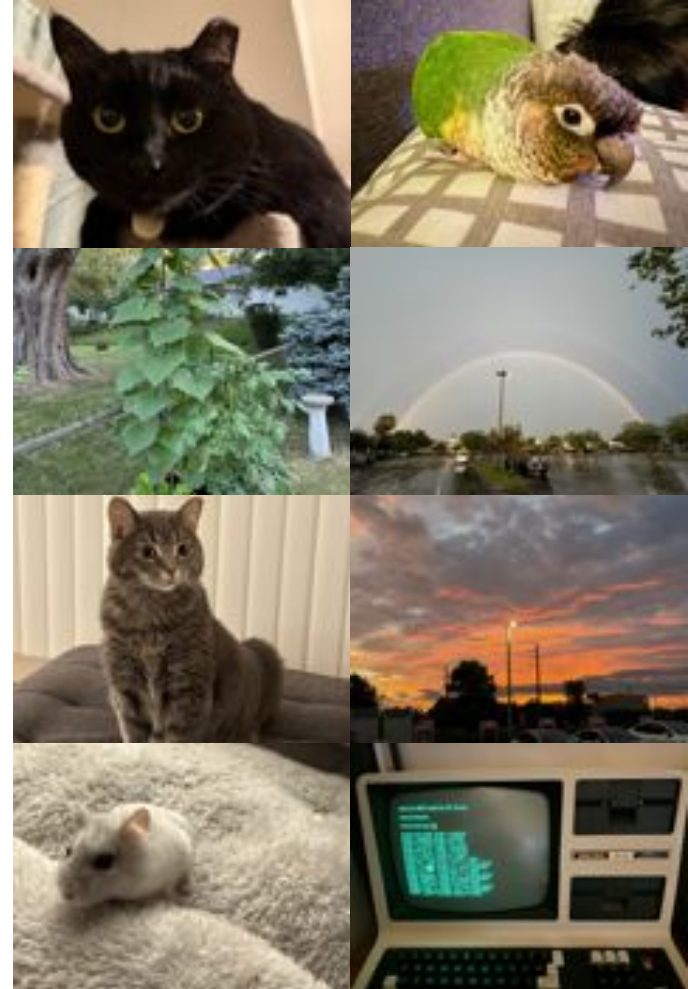
- We will compare the inherent **performance** of the techniques by measuring the overhead of the techniques and comparing the original and decompressed images.
- We will also compare the robustness of the techniques when the encodings are sent over a **lossy network** by examining how the loss and switching of packets affects the decoding process.

Approach - Algorithms

- **JPEG:** A lossy image compression technique that leverages Discrete Cosine Transforms.
- **K-Means:** A lossy compression technique clusters pixels into one of K colors.
- **(Modified) Run-Length Encoding:** The originally lossless compression technique that conveys the lengths of sequences of the same pixel has been modified to be lossy and now considers two pixels equal if the Euclidean distance between their color values is within a threshold.

Validation - Metrics and Images

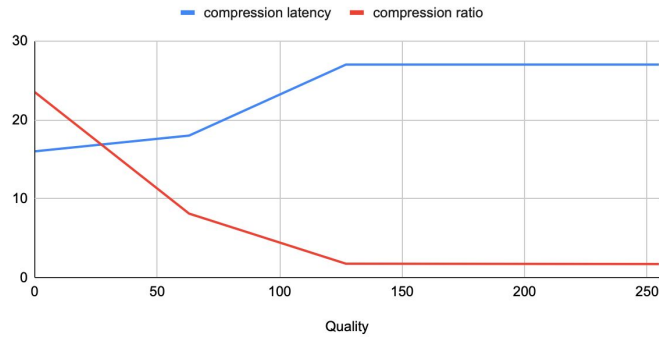
- **Compression Metrics:**
 - **Compression Latency:** Time taken to compress an image.
 - **Compression Ratio:** The size of the original image divided by the size of the compressed image.
- **Qualitative Metrics:**
 - **Mean Square Error:** Average squared difference between the pixels of the original and decompressed images.
 - **Peak Signal-to-Noise Ratio:** 10 times the base-10 logarithm of the maximum possible image pixel value squared divided by the MSE.
 - **Structural Similarity Index:** Evaluates perceived similarity of structure between images.



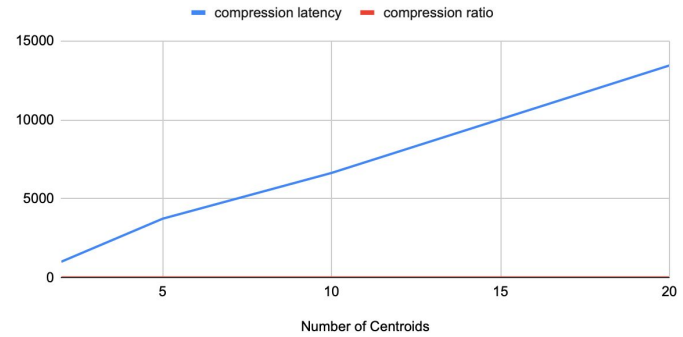
Images Used for Experiments

Validation - Preliminary Results (Compression Metrics)

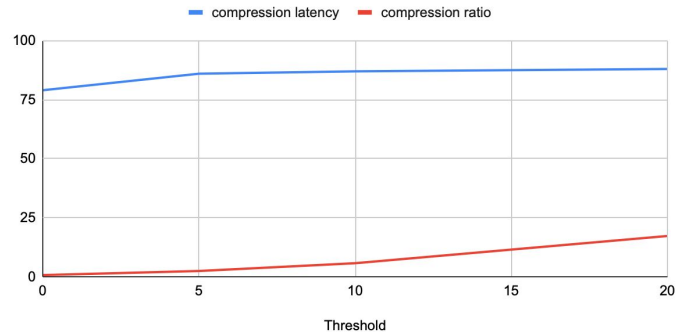
Influence of Hyperparameters on JPEG (compression ratio and latency)



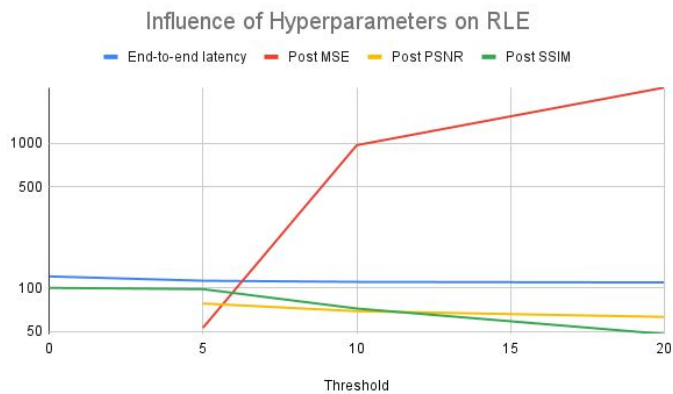
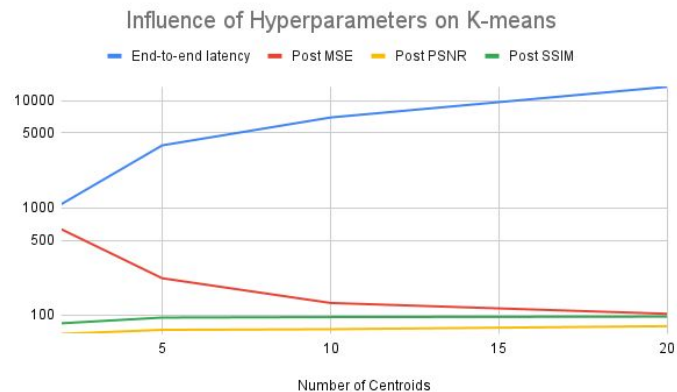
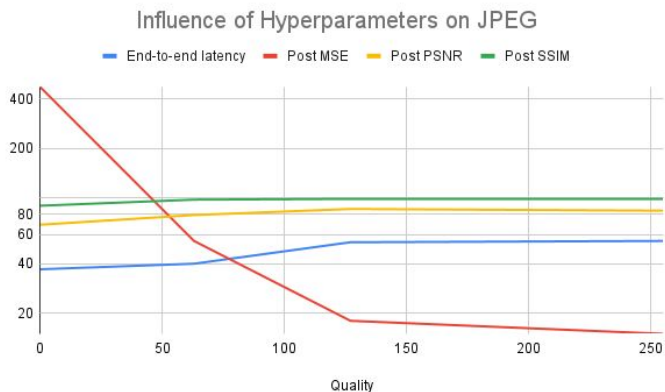
Influence of Hyperparameters on K-means (compression ratio and latency)



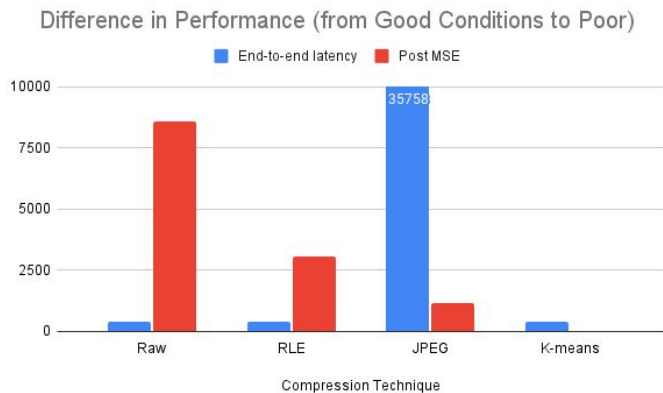
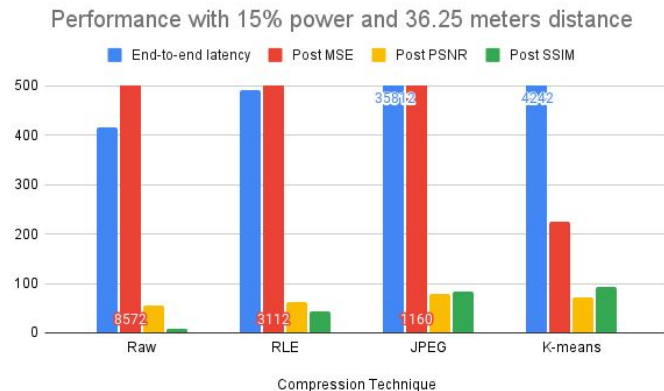
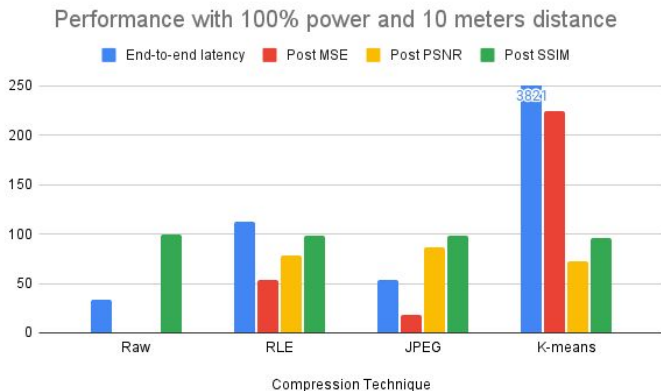
Influence of Hyperparameters on RLE (compression ratio and latency)



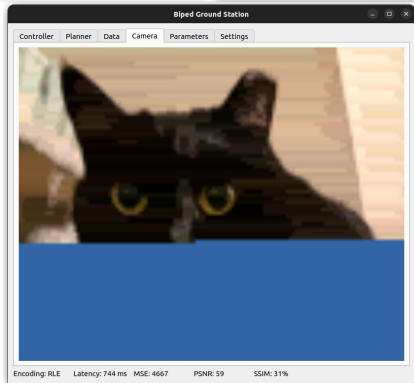
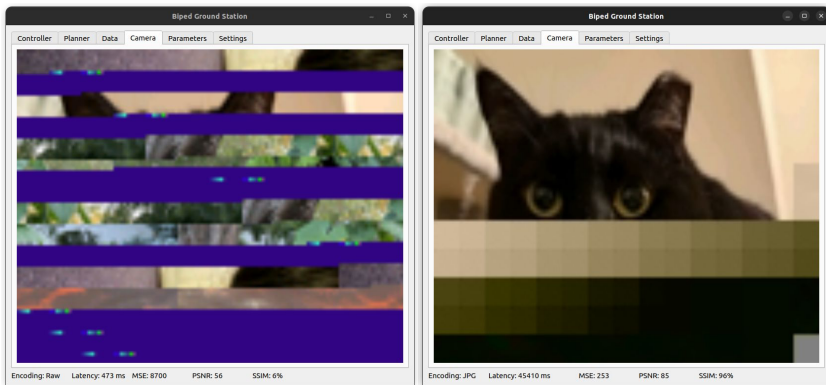
Validation - Preliminary Results (Qualitative Metrics)



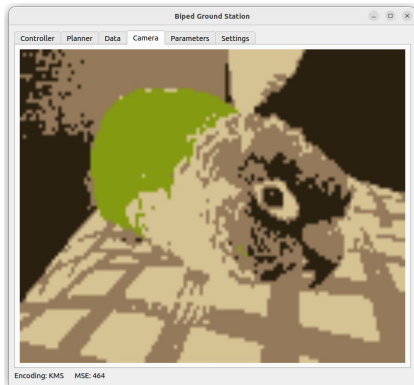
Validation - Preliminary Results (Lossy Network)



Validation - Preliminary Results (Images)



Corrupted Images



K-Means Output

```
/bash
qt.gui.imageio.jpeg: Corrupt JPEG data: premature end of data segment
qt.gui.imageio.jpeg: Invalid JPEG file structure: two SOI markers
qt.gui.imageio.jpeg: Corrupt JPEG data: premature end of data segment
qt.gui.imageio.jpeg: Invalid JPEG file structure: two SOI markers
qt.gui.imageio.jpeg: Corrupt JPEG data: 655 extraneous bytes before marker 0x8d
qt.gui.imageio.jpeg: Invalid JPEG file structure: two SOI markers
qt.gui.imageio.jpeg: Corrupt JPEG data: premature end of data segment
qt.gui.imageio.jpeg: Invalid JPEG file structure: two SOI markers
qt.gui.imageio.jpeg: Corrupt JPEG data: premature end of data segment
qt.gui.imageio.jpeg: Invalid JPEG file structure: two SOI markers
qt.gui.imageio.jpeg: Corrupt JPEG data: 751 extraneous bytes before marker 0x8d
qt.gui.imageio.jpeg: Invalid JPEG file structure: two SOI markers
qt.gui.imageio.jpeg: Corrupt JPEG data: premature end of data segment
qt.gui.imageio.jpeg: Invalid JPEG file structure: two SOI markers
qt.gui.imageio.jpeg: Corrupt JPEG data: premature end of data segment
qt.gui.imageio.jpeg: Invalid JPEG file structure: two SOI markers
qt.gui.imageio.jpeg: Corrupt JPEG data: 116 extraneous bytes before marker 0x8d
qt.gui.imageio.jpeg: Invalid JPEG file structure: two SOI markers
qt.gui.imageio.jpeg: Corrupt JPEG data: premature end of data segment
qt.gui.imageio.jpeg: Invalid JPEG file structure: two SOI markers
qt.gui.imageio.jpeg: Corrupt JPEG data: premature end of data segment
qt.gui.imageio.jpeg: Invalid JPEG file structure: two SOI markers
qt.gui.imageio.jpeg: Corrupt JPEG data: 1613 extraneous bytes before marker 0x8d
qt.gui.imageio.jpeg: Invalid JPEG file structure: two SOI markers
qt.gui.imageio.jpeg: Corrupt JPEG data: premature end of data segment
qt.gui.imageio.jpeg: Invalid JPEG file structure: two SOI markers
qt.gui.imageio.jpeg: Corrupt JPEG data: premature end of data segment
qt.gui.imageio.jpeg: Invalid JPEG file structure: two SOI markers
qt.gui.imageio.jpeg: Corrupt JPEG data: 783 extraneous bytes before marker 0x8d
qt.gui.imageio.jpeg: Invalid JPEG file structure: two SOI markers
qt.gui.imageio.jpeg: Corrupt JPEG data: premature end of data segment
qt.gui.imageio.jpeg: Invalid JPEG file structure: two SOI markers
qt.gui.imageio.jpeg: Corrupt JPEG data: 3508 extraneous bytes before marker 0x8d
qt.gui.imageio.jpeg: Invalid JPEG file structure: two SOI markers
qt.gui.imageio.jpeg: Corrupt JPEG data: premature end of data segment
qt.gui.imageio.jpeg: Invalid JPEG file structure: two SOI markers
qt.gui.imageio.jpeg: Corrupt JPEG data: 2180 extraneous bytes before marker 0x8d
qt.gui.imageio.jpeg: Invalid JPEG file structure: two SOI markers
qt.gui.imageio.jpeg: Corrupt JPEG data: premature end of data segment
qt.gui.imageio.jpeg: Invalid JPEG file structure: two SOI markers
```

JPEG Corruptions

Conclusion

- JPEG is efficient in terms of compression ratio and latency but fails to handle packet drops under lossy networks.
- RLE is slightly less efficient compared to JPEG but handles packet drops well in lossy network conditions.
- K-Means has a near static compression ratio but is extremely computationally heavy and has the worst qualitative metrics.