Implementation of Simultaneous Multi-Streaming of Live Solar Eclipse Video via 5.8 GHz AirMax

Tenger Batjargal[1], Wookwon Lee[2], and Nicholas B. Conklin[3]
Gannon University, Erie, PA 16541

For the 2017 solar eclipse ballooning, we have developed a video payload that can simultaneously live-stream multiple videos via a single 5.8 GHz wireless link through a Ubiquiti Rocket M5 modem. In this paper, we describe our approach to multiplex multiple video streams into a single data stream that a 5.8 GHz wireless modem can transport to the ground station. Various key factors are described to properly configure the Raspberry Pis and optimize the transmission from an M5 modem on the video payload over the 5.8 GHz link while ensuring adequate range and acceptable video quality received at the ground station. A screenshot of the multi-video streaming is provided as an example to justify a successful operation of our video payload.

I. Introduction

FOR the 2017 solar eclipse ballooning, the live streaming of the videos capturing the shadows of the solar eclipse was one of the main objectives. All teams who participated in the 2017 solar eclipse ballooning project were initially provided with a baseline ballooning system [1][2] that consisted of a ground station and four standard payloads for still image, video, Iridium-based tracking, and cut-down. In particular, the standard video payload for live streaming was primarily composed of a Raspberry Pi 2 Model B board with a Pi-camera module to capture high-definition video during balloon flight and a 5.8 GHz Ubiquiti Rocket M5 modem to transmit the video to its counterpart M5 modem on the ground station. The Pi-camera was installed on a servo motor to be able to adjust its viewing direction to some extent but its primary viewing angle was still toward only one of the four directions, i.e., north, south, east, and west of the payload position. To overcome this drawback and be able to capture video images from more than one particular direction, there were other approaches adopted by some other solar eclipse ballooning teams in the nation, such as placing the Pi-camera on a servo motor on top of the video payload, developing a video selector board that could select, at a given time, one of the video streams from multiple Pi-cameras facing different directions, or replicating multiple Raspberry Pi boards, each with a Pi-camera and a dedicated M5 modem to simultaneously transmit all live video streams from multiple Pi-cameras on multiple 5.8 GHz modems to the ground for further processing. Each of these approaches presents pros and cons in achieving the objective of capturing the solar eclipse images live from multiple viewing directions but the analysis of such pros and cons is omitted herein as it falls beyond the scope of this paper.

Different from the approaches mentioned above, we have developed a video payload that can simultaneously live-stream multiple videos via a single 5.8 GHz wireless link. The key design aspects we explored and ultimately adopted were 1) using more than one camera on a Raspberry Pi board and 2) multiplexing all video streams from multiple cameras onto a single stream of data traffic that a single 5.8 GHz radio modem can transmit to the ground station. With much troubleshooting and careful optimization of all components' functionality for this method, we have finally chosen to use 4 Raspberry Pis, each with an SD memory card and two cameras (a Pi-camera and a webcam), and a high-bit-rate network switch, as well as a power-over-Ethernet (POE) device and an M5 AirMax modem transmitting through two rubber-duck antennas as originally included in the standard video payload. As our design objective for the video payload was to cover four non-overlapping directions of north, south, east, and west from the payload orientation, only four of those eight cameras were used for live streaming while all video streams from the eight cameras were stored on the on-board memory cards.

[1] Undergraduate Student, Department of Electrical and Computer Engineering, Gannon University, Erie, PA 16541.
[2] Associate Professor and Chair, Dept. of Electrical and Computer Engineering, Gannon University, Erie, PA 16541.
[3] Associate Professor and Chair, Physics Program, School of Sciences, Gannon University, Erie, Pennsylvania 16541.

In this paper, we present technical details of our video payload implementation. Section II presents an overview of the system. In Section III, technical details of simultaneously operating two cameras on a single Raspberry Pi are first provided. Then, as one of the key aspects of our approach, we describe how multiplexing of multiple video streams into a single data stream was achieved such that a single M5 modem could transport all video streams to the ground station through the 5.8 GHz link. Various key factors are also discussed in optimizing the wireless transmission from the M5 modem on the video payload while ensuring acceptable quality of the live video at the ground station.

II. Overview of the System

An overall functional block diagram of our video payload is shown in Fig. 1 along with its counterpart in the ground station. As shown in the diagram, on the payload side, each of the four Raspberry Pis is assigned a static IP address sequentially from 105 to 108, i.e., 192.168.1.x, and integrates one webcam and one Pi-camera. The webcam is connected to one of the four USB ports of the Raspberry Pi and the Pi-camera was connected to the Pi-camera serial interface slot available on the Raspberry Pi (version 2, Model B). For data communication, each Raspberry Pi is connected to an Ethernet switch via a category-6 flat Internet network cable with RJ45 connectors on both ends.

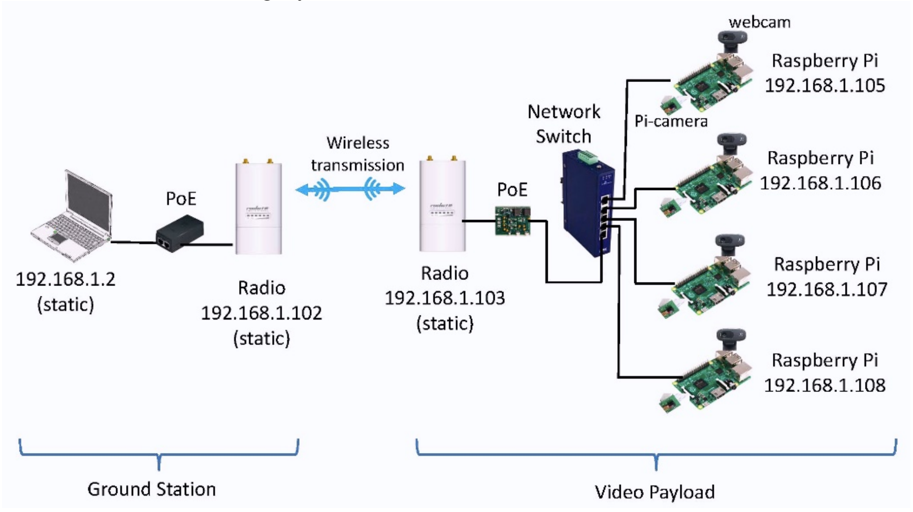


Fig. 1 Functional block diagram of the multi-streaming video payload.

The Ethernet switch facilitates data flow between the radio modem with a static IP address (e.g., 192.168.1.103) and four Raspberry Pis. The Ethernet switch we have chosen supports five 10/100/1000 Base-T Ethernet ports and thus each port is compatible with the 10/100 Base-T Ethernet port available on the Raspberry Pi board. Four Ethernet ports are used for data flow to/from four Raspberry Pis, respectively, and the 5th Ethernet port is for data flow to/from the Power-over-Ethernet (PoE) board which was a part of the standard video payload initially provided to us. As the Ethernet switch must be able to function properly over a wide range of temperatures for the high-altitude ballooning purposes, we have chosen an industrial Ethernet switch [3] that can operate in a temperature range of -40° to 167°F (-40° to 75°C). Note that typical commercial Ethernet switches operate in a narrower temperature range such as 32° to 122°F (0° to 50°C). There are other industrial Ethernet switches that support up to 8 Ethernet ports which could in turn facilitate data flow between the radio modem and up to 8 Raspberry Pis. Also, in principle, more than one Ethernet switches could be connected in a cascade to facilitate data from to/from any number of Raspberry Pis. However, considering various factors that are identified in the following section, we had finalized our video payload with four Raspberry Pis and thus a single five-port Ether switch was a suitable choice.

The radio modem is a 5.8 GHz Ubiquiti modem, the Rocket M5 [4], as included in the standard video payload and paired with a PoE device shown in the block diagram. In the following section, we further describe our troubleshooting and configuration to optimize the operation of the M5 modem for simultaneous live streaming of videos from four Raspberry Pis.

On the ground station, another pair of an M5 modem and a PoE device is used as originally installed on the baseline ground station system. All parameters of the M5 on the ground station except for one are set the same as those of the M5 on the payload. We will discuss this further as well in the next section. Then, the PoE is connected to a laptop computer on which a software application is run to display four streaming videos. We have used both VLC media player and Open Broadcaster Software (OBS). VLC can only display a single stream in a window and thus multiple

VLC windows are necessary for simultaneous multiple streaming. OBS can be configured to display multiple streaming videos on a single window, and thus it was our choice to upstream simultaneously four live-streaming videos from our video payload onto the online repository site during the August 21, 2017 solar eclipse ballooning.

III. Details of Subsystem Configuration

A. Video Streaming with Two Cameras on a Single Raspberry Pi

The design requirement we initially set was to keep the Pi-camera as provided in the standard video payload [2] but add more cameras. As there is only one Pi-camera slot on a Raspberry Pi, this requirement led to using an additional webcam through one of the four USB ports available on the Raspberry Pi. One challenge was the encoding and processing of multiple video streams in the Raspberry Pi mainly due to the large amount of data for video streaming and the limited bandwidth of the M5 modem. The H.264 encoder was used to compress and process the video from the Pi camera as this encoding scheme can deliver high-definition video at a reduced data rate. For the webcam video, we adopted the commonly used, Motion-JPEG (MJPEG) scheme to encode and process the video as it was relatively simpler to set up compared to the H.264 encoding. The steps shown in Fig. 2 and Fig. 3 were taken to establish video streaming with two cameras. Note that, for the radio modem in this development, we utilized an M2 modem, which is from the same vendor and has the same functionality as the M5 except that it operates on the 2.5 GHz band. Both M2 and M5 are configured by the same software, airOS [6].

I. Setting up the IP address

1. Type ***sudo nano /etc/dhcpcd.conf*** on Raspberry Pi.
2. Press Enter. Then, go all the way to the end of the content, and type
   - **interface eth0**
   - **static ip_address=192.168.1.107/24**
   - **static routers=192.168.1.20**
   - **static domain_name_servers=192.168.1.20**
3. To exit the editor, press **ctrl+x**
4. To save the changes press the letter "**Y**" then hit **Enter**
5. Press **sudo reboot**
6. After the device is rebooted, type **sudo ifconfig** to check if the IP address has been successfully changed.
7. To check the connection, **ping** the given IP address, e.g., 192.168.1.107 from the ground station laptop.

II. Download the Motion Service

*** Connect the Raspberry Pi to the Internet:**

1. Connect the Raspberry Pi to the Internet through the Ethernet port
2. Comment out the 3 lines of assigning a static IP address by ***sudo nano /etc/dhcpcd.conf***
3. To exit the editor, type **ctrl+x**
4. To save the changes press the letter "**Y**" then hit **Enter**
5. Type **sudo reboot**
6. **Login: pi**
7. **Password: raspberry**
8. Type **sudo ifconfig** to check the IP address, which supposed to be allocated from the server.

2. Download the Motion Service (cont.)

*** Download the Motion Service**
http://www.instructables.com/id/How-to-Make-Raspberry-Pi-Webcam-Server-and-Stream-/

1. Type **sudo apt-get update** just to update the tool
2. Type **sudo apt-get install motion** (it takes 1-2 minutes to download the Motion package; wait until the whole process finishes)
3. Type **lsusb** to see name of the camera. If it is NOT there, then there is some problem with the camera or the camera is not supported in the Motion package.

3. Configure the Motion Service

1. Press **sudo nano /etc/motion/motion.conf**
2. Make sure **'daemon'** is **ON**.
3. Set **'framerate'** anywhere between **1000 to 1500**.
4. Set '**width**' & '**height**' to **640** & **480.**
5. Keep **Stream port** to **8081.**
6. **webcam_quality** should be **100.**
7. Change **webcam_localhost** to **OFF.**
8. Change '**webcontrol_localhost**' to **OFF.**
9. Set **post_capture** to **5**
10. Type **ctrl + x** to exit. Type **y** to save and press **Enter** to confirm.
11. Type **sudo nano /etc/default/motion**
12. Set **start_motion_daemon** to **yes.**
13. Type ***sudo nano /etc/dhcpcd.conf*** to uncomment the 3 static lines
14. Connect the Raspberry Pi's Ethernet cable back to M2 or M5 modem.
15. Type **sudo reboot** to restart the Pi (This also restart the motion service)
    a. Type **sudo service motion restart** (in case Motion does not start)
16. On the ground station laptop, open a web browser and point to **192.168.1.107:8081.**

Fig. 2 Steps to configure the Raspberry Pi for a webcam.

1. Configure the *stream.sh*

Since the Motion Service also tries to use port 8080 as its one of the default feature, the port number for the Pi-cam should be changed to a different one.

1. Login to the Raspberry Pi
2. Type **cd Ubiquiti_Pi_Code** (full path: **/home/pi/Ubiquiti_Pi_Code)** When logged in, the user is in **/home/pi**.

3. Type **sudo nano stream.sh**
4. Change the port number **8080** to **8888.**
5. To exit the editor, type **ctrl+x.**
6. To save your changes type **'Y'** and hit **Enter.**
7. Type **./stream.sh** to start video streaming on the Raspberry PI.
8. On the ground station laptop, open VLC, go to **Open Network Stream** to enter the IP address and port number.

Fig. 3 Steps to configure the Raspberry Pi for a Pi-camera.

While two sources of video streaming were available on each Raspberry Pi, we have chosen to stream only the Pi-camera primarily due to the limited transmission capacity of the radio modem and also the amount of data required for video streaming from the two sources of video, which are two of the key factors for the quality of the video received and displayed at the ground station. We observed that, for the video streaming, the central processing unit (CPU) on the Raspberry Pi re-directs the raw video input from the Pi-camera to its Dual VideoCore IV® Multimedia Co-Processor functioning as a graphics processing unit (GPU) and thus, is less CPU processing-intensive than the CPU processing required for the raw video input from the webcam. With the Pi-camera used for live streaming, the Raspberry Pi consumed only 6-8% of its CPU processing power and in turn was able to facilitate other processing better such as keeping the communication session between the Raspberry Pi (e.g., 192.168.1.105) on the video payload and the computer (i.e., 192.168.1.2) at the ground station. On the other hand, for the webcam, we have integrated a Microsoft LifeCam HD-3000. Since the USB ports on the Raspberry Pi are not connected to the graphics processing unit, the processing of the encoding and transmission of the video is done by the CPU itself. We observed that general video recording and streaming consumed around 65 – 90% of the CPU processing power of the Raspberry Pi. This heavy use of the CPU processing power often resulted in a loss of communication session between the Raspberry Pi and the computer at the ground station.

To optimize the quality of the video at the ground station, after much troubleshooting, we have set the image resolution, bitrate, and framerate for the images from the Pi camera in a shell script "stream.sh" as follows:

stream.sh for the standard video payload	**stream.sh** for our optimized video
• resolution = 1920 x 1080 • framerate = 25 fps • bitrate = 2,000,000 bps	• resolution = 640 x 380 • framerate = 15 fps • bitrate = 500000

Fig. 4 Key parameters for video from the Pi-camera.

Note that the framerate for the standard video payload was 25 frames per second. In order to decrease the bandwidth requirements, we reduced this to 15 frames per second. We noticed a great deal of blocking/blurriness when the payload was moving quickly, so we inserted i-frames at 15 frame intervals (instead of the default 250). I-frames, contain full image data, as opposed to p-frames, which are based on the difference between the previous frame and the current frame. Although i-frames are less compressible and increase the amount of data that must be throughput, the shorter p-frame interval significantly reduced blocking during fast motion. These changes allowed us to maintain adequate video quality, even while the payload was in motion.

Also, it is worthwhile mentioning some observations we made during the troubleshooting for the webcam. As noted earlier, we installed and began troubleshooting with the Motion package and then later with ffmpeg for the key parameters to optimize video quality at the ground station, such as resolution, framerate, and bitrate. Unfortunately, our observations revealed that the Raspberry Pi's processing of webcam videos required a high data rate of 45-64 Mbps to deliver the streaming video to the ground station without blurriness when the payload is in motion but the transmission rate of the M5 radio used for the video payload was a bottleneck. As noted further below, the M5 offers several options of transmission data rates but the data rates are also tied to the transmission distance (i.e., range). When the M5 is set to MCS-5, which supports a data rate of 4.8 Mbps, its line-of-sight range estimated was about 20 miles. As the range of 20 miles was the minimum desired for the solar eclipse ballooning with the ground station fixed at a location, other options of M5 offering a higher data rate couldn't be used as those options would reduce the range substantially, e.g., a few miles. Another radio we had, i.e., M2, offers a different set of options and supports well the video streaming with a webcam as it can be configured to MCS-15 which supports a data rate of 270/300 Mbps. As we desired to use the M5 that operates on the less crowded 5.8 GHz band than 2.5 GHz band, we dropped the idea of streaming with the webcam and decided to record the videos and save them locally on the SD card installed on each Raspberry Pi. For the recording of the webcam video, a higher resolution was set to resolution=1280x760, framerate=10, and the video was saved every 10 minutes of duration using a new script written with avconv, which is freely available online [7].

B. High Bit-Rate Network Switch for Multiplexing of Video Streams

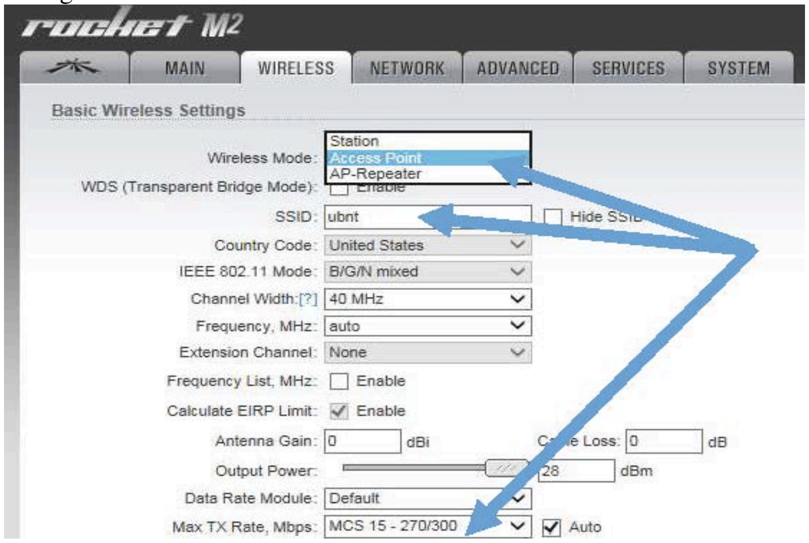
Each of the four Raspberry Pis for flight is configured to simultaneously process two video streams as described in the previous subsection. They are connected to the Ethernet switch as illustrated in Fig. 1, and no further configuration on this switch is necessary.

One thing to note is that a large amount of data is processed through the Ethernet switch since the switch gets the input from the four video sources and outputs to the radio modem. As a result, the Ethernet switch generates quite a bit of heat. Also, in high-altitude ballooning, passive or active cooling via air circulation is impossible and the payload is typically sealed. As such, it is critical to use a wide temperature-range industrial switch. Obviously, if the Ethernet switch malfunctions or completely fails, the entire mission of the video streaming fails for the ground station.

C. Key Parameters for 5.8 GHz AirMax Modem Configuration

The M5 AirMax modem is configured with an online interface called the *airOS Configuration Interface* provided by the device vendor [6]. The online interface can be accessed at https://192.168.1.20 once the modem is connected via an Ethernet cable to a laptop and the laptop is configured with a static IP address on the 192.168.1.x subnet, e.g., 192.168.1.2 for our case (see Fig. 1). One can easily follow the set-up guide to set key configuration parameters such as Wireless Mode, SSID, Max TX Rate (under the Wireless tab), and IP address (under the Network tab). An example is shown in Fig. 5. For the M5 on the ground station, the Wireless Mode parameter is set to *Access Point* while the M5 on the video payload is set to *Station*. Their static IP addresses must be unique on the 192.168.1.x subnet (see Fig. 1 for our case).

As mentioned earlier, the M5 offers several options of maximum transmission data rate (for example, see *Max. Tx Rate* parameter in Fig. 5) and these transmission rates, ranging from MCS-0 to MCS-15, are explicitly mapped to their respective Tx and Rx power specification [8]. For example, for MCS-5 of M5 with 802.11n/AirMax modulation, the average Tx power is 24 dBm and Rx sensitivity is -83 dBm. Transmission power and receiving sensitivity are key parameters effecting the range over which the wireless link can be maintained.



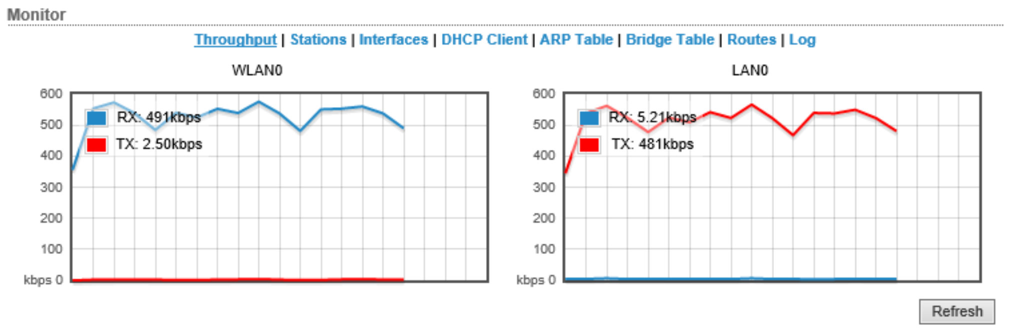
Basic Wireless Settings

Wireless Mode: Station / Access Point / AP-Repeater
WDS (Transparent Bridge Mode): Enable
SSID: ubnt Hide SSID
Country Code: United States
IEEE 802.11 Mode: B/G/N mixed
Channel Width:[?] 40 MHz
Frequency, MHz: auto
Extension Channel: None
Frequency List, MHz: Enable
Calculate EIRP Limit: ✓ Enable
Antenna Gain: 0 dBi Cable Loss: 0 dB
Output Power: 28 dBm
Data Rate Module: Default
Max TX Rate, Mbps: MCS 15 - 270/300 ✓ Auto

Fig. 5 Examples of key M5 parameters.

D. Exemplary Testing of M5 for Video Streaming

In this section, we provide examples of M5 parameter settings and observations of actual throughput, and its impact on video quality. For this purpose, we set the Max TX Rate "MCS-5, 19.5 Mbps" under the Wireless tab. For the video streaming with four Pi-cameras, Fig. 6 shows the Tx/Rx data rates measured at the ground station M5 modem with the Ethernet port receiving data from the payload M5 as WLAN0 in the figure and the Ethernet port transmitting to the laptop as LAN0. As one can see, the ground station M5 modem receives data at an Rx rate of 491 kbps and transmits to the laptop at a Tx rate of 481 kbps. Over the course of observation, we noted that the data rate ranges from ~450 kbps to ~550 kbps. Its implications on video quality is briefly summarized in Table 1.



(a) Receiving data rate, 491 kbps (b) Transmitting data rate, 481 kbps

Fig. 6 Receive and transmit data rates for streaming video from four Pi-cameras

Table 1 Examples of video quality received via M5 from Pi-camera

	Video Quality (0 – 10 with 10 for the best quality)	**with/without payload movement**
4 Pi Camera	9, smooth and almost no blurriness but some delays between video streams from different Pi-cameras	Without Movement
4 Pi Camera	3, not smooth and blurry	With Movement

The Tx/Rx data rates in the same M5 configuration except that video streams are generated by four webcams is shown in Fig. 7. One can easily note that the ground station M5 modem receives data at an Rx rate of 3.13 Mbps and transmits to the laptop at a Tx rate of 3.09 Mbps. Over the course of observation, we noted that the data rate ranges from ~2.5 Mbps to ~3.2 Mbps. Its implications on video quality is briefly summarized in Table 2.

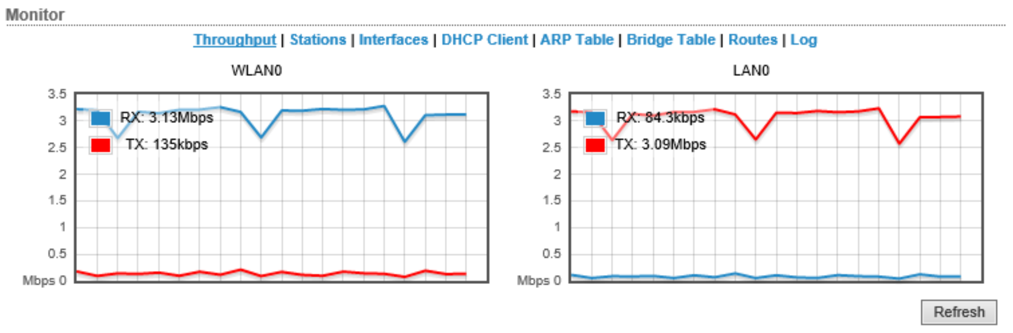
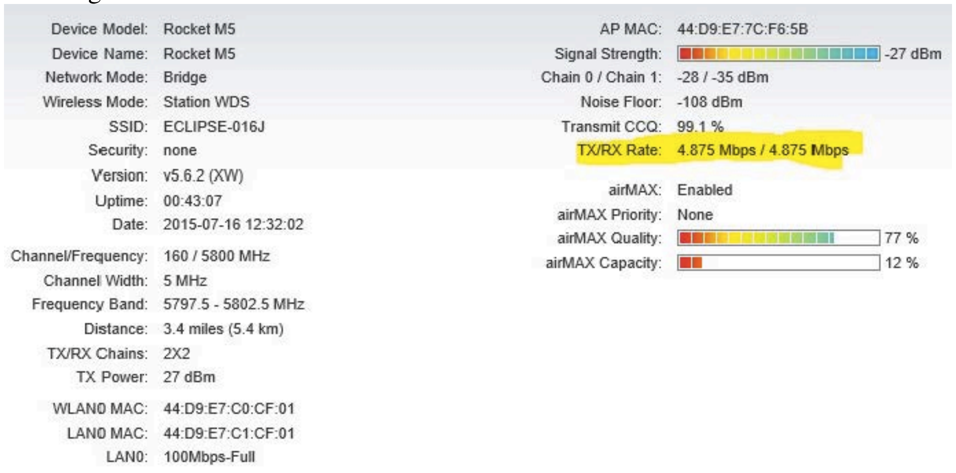


Fig. 7 Receive and transmit data rates for streaming video from four webcams

Table 2 Examples of video quality received via M5 from four webcams

	Video Quality (0 – 10 with 10 for the best quality)	**with/without payload movement**
4 Web Camera	1, not smooth and almost no blurriness	Without Movement
4 Web Camera	1, not smooth and almost no blurriness	With Movement

Shown in Fig. 8, we also observed that the Max. Tx/Rx Rate of M5 was displayed to be 4.875 Mbps for MCS-5 under the Main tab of airOS interface. Considering this displayed data rate as the max raw data rate, an effective data rate (or throughput) was ~3 Mbps as mentioned above for MCS-5 over a 5 MHz bandwidth setting. For further testing, when only one webcam is set to stream, the M5 receive data rate was immediately hitting 2.5 Mbps ~ 3.2 Mbps. However, not shown in this paper due to the space limitation, when an M2 is used and set to a much higher Max Tx/Rx data rate, e.g., 90 Mbps, the M2 didn't experience any problem of processing an aggregated data rate of ~64 Mbps from all four webcams. From these, we deduce that the M5's MCS-5 option cannot support live streaming videos from four webcams using an MJPEG encoder. Even further, the received images were somewhat discontinuous and appeared as if the images were taken at discrete time instants.



```
     Device Model:  Rocket M5                         AP MAC:  44:D9:E7:7C:F6:5B
      Device Name:  Rocket M5                Signal Strength:  -27 dBm
     Network Mode:  Bridge                 Chain 0 / Chain 1:  -28 / -35 dBm
    Wireless Mode:  Station WDS                  Noise Floor:  -108 dBm
             SSID:  ECLIPSE-016J               Transmit CCQ:  99.1 %
         Security:  none                        TX/RX Rate:  4.875 Mbps / 4.875 Mbps
          Version:  v5.6.2 (XW)
           Uptime:  00:43:07                          airMAX:  Enabled
             Date:  2015-07-16 12:32:02       airMAX Priority:  None
                                               airMAX Quality:  77 %
Channel/Frequency:  160 / 5800 MHz            airMAX Capacity:  12 %
    Channel Width:  5 MHz
   Frequency Band:  5797.5 - 5802.5 MHz
         Distance:  3.4 miles (5.4 km)
    TX/RX Chains:  2X2
         TX Power:  27 dBm
        WLAN0 MAC:  44:D9:E7:C0:CF:01
         LAN0 MAC:  44:D9:E7:C1:CF:01
             LAN0:  100Mbps-Full
```

Fig. 8 Main configuration screen of M5 with MCS-5 data rate (19.5 Mbps)

Our various tests on the M5 modem led us to conclude that only the Pi-cameras should be used for live video streaming for our solar eclipse ballooning project. We were able to optimize key parameters for the Pi-camera as given in Fig. 4 above. We were able to simultaneously receive data from all four cameras during the balloon ascent before the payload went out of range. Fig. 9 below shows a screen shot of the OBS streaming during the August 21, 2017 flight.

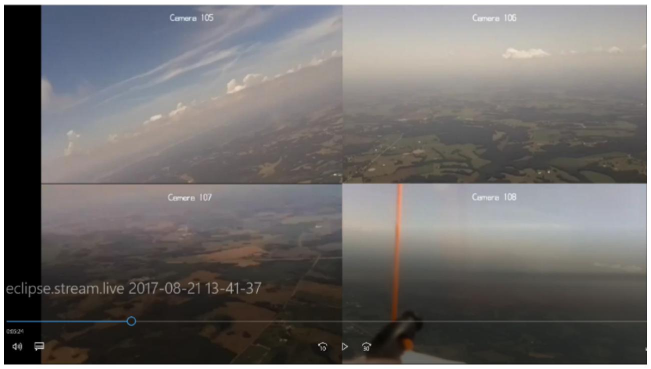


Fig. 9 Screenshot of four simultaneous streaming during the Aug. 21, 2017 flight

IV. Conclusions

We have described how we achieved simultaneous multiple video streaming using four Pi-cameras and an Ethernet switch over a single 5.8 GHz wireless link between the video payload and our ground station. We believe that our approach is new to the high-altitude ballooning community and we hope that it will provide the ballooning community with an improved video streaming method that can be enjoyed over many years to come.

Acknowledgments

This work was supported in part by the Pennsylvania Space Grant Consortium and in part by the Minnesota Space Grant Consortium's Eclipse Ballooning Project. We thank Andrew Wee and Kaiguang Jiang, visiting undergraduate students from Nanyang Polytechnic (NYP), Singapore, in spring 2017 for their initial work of integrating two cameras onto a single Raspberry Pi. We are grateful to other student members of the Gannon ballooning team who participated in numerous field integration-testing of our solar eclipse ballooning system.

References

[1] Lee, W. and Conklin, N. B., "Solar Eclipse Ballooning with a Multiband Tracking Subsystem for Undergraduate Research Experience," in Proc. ASEE Annual Conference and Exposition, June 25 - 28, 2017, Columbus, Ohio, pp. 1-11.

[2] Eclipse Ballooning Project: About the Eclipse Ballooning Project Systems, on line, last accessed on Oct. 8, 2017 at http://eclipse.montana.edu/about-the-systems/.

[3] B&B Electronics Mfg. Co. Inc., *ELNIX Industrial Ethernet Switch EIR405-T: User Manual*, Doc. Number: EIR405-T – 0912m, 2008.

[4] Ubiquiti Networks, Quick Start Guide for Rocket M Carrier Class airMAX Base Station, on line, last accessed on Oct. 8, 2017 at https://dl.ubnt.com/guides/Rocket_M/RocketM_Series_QSG.pdf.

[5] Eclipse Ballooning Project, online, last accessed on Oct. 8, 2017 at http://eclipse.montana.edu/.

[6] Ubiquiti Networks, *airOS 5 User Guide*, release version 5.6, 2015, on line, last accessed on Oct. 8, 2017 at https://dl.ubnt.com/guides/airOS/airOS_UG.pdf.

[7] *avconv Documentation*, on line, last accessed on Oct. 8, 2017 at https://libav.org/avconv.html.

[8] Ubiquiti Networks, *Data Sheet for Rocket M Powerful 2x2 MIMO airMAX Base Station*, on line, last accessed on Oct. 8, 2017 at https://dl.ubnt.com/datasheets/rocketm/RocketM_DS.pdf.