System Architecture of a High Altitude Live-Streaming System

Nikhil Nair, Matthew Wise, Natalie Condzal, Ruben Turner, Sanya Doda
University of Maryland, College Park

I. Introduction

In the late 1960s, high altitude airships were used for real-time communications experimentation in a high altitude environment; however, this high altitude research platform has not been readily maintained as time has progressed [1]. While many live-streaming systems have been developed for real-time surveillance purposes, there has not been much surveillance development work performed using a high altitude ballooning platform in recent times. By utilizing University ballooning programs to perform this live-streaming research, the research and development phase can be explored at a lower cost while continually achieving high level results.

The same capabilities desired in a real-time streaming system are desired in a surveillance setting due to the importance of a high data rate communications link that is key for many applications such as emergency and military. The concepts employed by a surveillance system are similar to the ones proposed in this paper and will enable both research and industrial enterprises to utilize this commercial off the shelf (COTS) system for a baseline surveillance system architecture.

II. Abstract

Modern methods of surveillance can be achieved through various methods including drones, satellite imagery, and aircraft surveillance. However, many of these methods have design flaws; most notably, these design flaws include power usage issues, detectability, and range of coverage. Through the use of a payload and ground station with dish and satellite hardware on a high-altitude balloon flight, one can live-stream aerial footage and greatly reduce security risks. Using a high altitude balloon as the basis for flight not only increases aerial range but also flight time since the batteries are only used to power the hardware required for streaming. The payload also enables lower package visibility as the payload would not be easily caught using radar or similar technologies. We propose a system with the overall goal of providing a reliable high data rate communications link for low latency telemetry and video throughout a high-altitude balloon flight. This system is composed of commercial off the shelf (COTS) parts including Ubiquiti hardware to ensure ease in duplication for industrial use. To ensure a multi-layered encryption and self-monitoring system, we would use two or more different languages to ensure constant, reliable, and encrypted communications on top of proficient rendering and visualization. This paper presents a detailed outline of our system architecture coupled with our test plan and preliminary results to ensure flight-readiness of our live-streaming system.

III. System Setup Procedure

The system is designed to involve two separate branches. The processing branch was a package that would take data input and both save it locally and broadcast it over a local network. The network branch took the locally broadcasted data and sent it to the ground station using a 5 gHz connection. A 2s Lithium-Polymer battery, providing 7.4v to the system was used to power the electronics. This battery holds 6.2 aH of power, providing a total 45.88 wH. This power is then converted using a Universal Battery Eliminator Circuit (UBEC) to provide a constant 5 volt 3 amp source for the processing branch. The networking branch used a 24 volt boost converter. The system can last at minimum about 2 hours and 50 minutes in non-ideal thermal conditions. In testing, the system was seen to have a 4 hour battery life, which is a safety factor above an average flight time [2]. A Low Voltage Cutoff (LVC) was used to cut power after the battery reached a certain voltage cutoff. This would reduce damage to our battery pack and increase overall battery health.

The Processing branch housed a Raspberry Pi 3b+ and a Teensy 3.5. The Teensy was attached to a BNO055 accelerometer and a metal gear servo. The BNO055 was flight proven for our Balloon Payload Program and was

used to reduce complexity and increase reliability. The data from the accelerometer was then used to control a metal gear servo. We had specifically chosen a metal gear servo in order to make sure that our proportional controller error correction was accurately achieved. The Teensy also relayed this information to the Raspberry Pi via a Universal Serial Bus (USB). USB was chosen over using GPIO pinouts in order to ensure fast and reliable data transfer while powering the Teensy with a continuous 5v power source and keeping overall weight down. The Raspberry Pi has 4 critical connections. The power is fed through the 5v rail in the GPIO. This enabled us to feed power directly from the UBEC by just using a connector. The Pi was also connected to the PiCam using the standard connector. As mentioned previously, the USB from the Teensy was plugged into the front ports on the Pi. An ethernet cable was plugged into the Pi from a Power over Ethernet adapter. This enabled the Pi to be seen on the local network.

The networking branch really only consisted of two parts. A PoE injector took network data from the Raspberry Pi and joined it with 24v power from the Boost converter. This data is then sent to the Ubiquiti Nanostation via an Ethernet Cable. In this whole package, the Nanostation draws the most power and produces the most heat. This means optimizing the rest of the hardware is critical. While not connected to the payload physically, the groundstation is also quite important. A Ubiquiti LiteBeam antenna is used to capture local data from the Nanostation using a 5 gHZ radio signal. While it may not be as powerful as other antenna choices, it paired well with the Nanostation as they were both running similar software suits. The Litebeam attached to a router via ethernet. The router created a Dynamic Host Configuration Protocol (DHCP) server list for all the devices connected on the network including the Litebeam, Nanostation, Pi, and Laptop. Finally, using the list to find the IP for the Raspberry Pi, we were able to get the video stream on a laptop. For a more secure application, we can use secure ports that have encryption to disable hacking. The video stream could also be directly encrypted [3]. In our application, we decided to stream to YouTube by using a hotspot connection to the laptop receiving the Local stream.

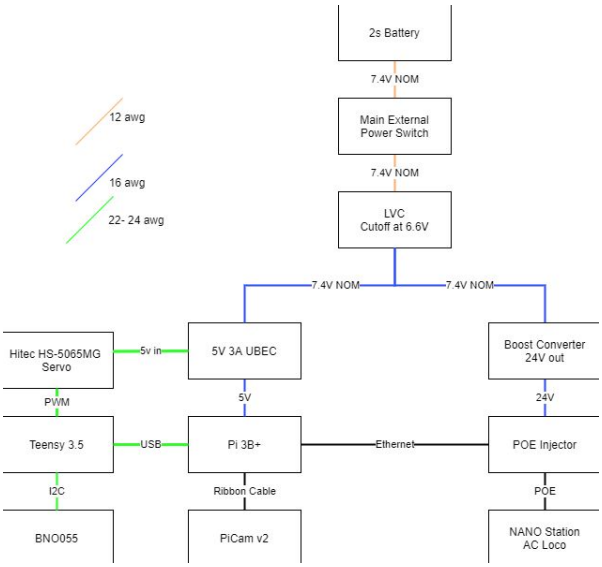

Figure 1. System Schematic

IV. Software Considerations

The software suite accomplishes two main objectives of maintaining a high data rate communication link throughout the flight and reliably logging data & video to be analyzed post-flight. The flight software suite has four main components: video streaming, video recording, proportional controller, and data logging. In addition, the flight software suite is supplemented with post-flight analysis software to process the large amounts of data received during flight and draw meaningful conclusions from the flight. The software was written using the version control software GitHub to support team-wide collaboration and quick development [4]. Additionally, both the flight and post-flight analysis software suites are open-source to allow for other ballooning programs to utilize the software in another live-streaming system.

The video recording and streaming software utilize the raspivid application, which supports video capture through a Raspberry Pi camera module. However, the native raspivid application that comes pre-loaded on the Raspberry Pi does not allow for simultaneous video recording and streaming while also saving the video feed as segmented clips. Modifications can be made to the open-source code for the application in order to allow for this additional functionality. With the modified version of the application, one can successfully record a stable video feed, save it in segmented clips for file loss and data corruption prevention, and stream the video feed to a local network, all at once. Other devices connected to the network would be able to view the stream, and, with a connection to the internet, this video feed could also be live streamed to Youtube for a broader audience.

PID (proportional-integral-derivative) controllers have historically been used to achieve stable video feed on unmanned aerial vehicles as well as other high precision applications [5]. For the live-streaming system architecture described in this paper, a proportional controller was used and will be upgraded to a PID controller in the future further error correction. The proportional controller software stabilizes the video feed from the Raspberry Pi camera module (PiCam) using a BNO055 absolute orientation sensor as well as the Arduino PID and BNO055 libraries. The proportional controller reads in the Z Euler angle to process the current pitch of the payload with respect to its home value. Next, the software leverages the proportional controller algorithm within the Arduino PID library to correct for the error between the setpoint and the current pitch angle. Through testing, a proportional gain value of 1.05 was chosen. Lastly, the proportional controller algorithm will output a servo value relative to the current home position of the payload to ensure that the PiCam maintains the same heading, thus resulting in a stable video feed. Note that the proportional controller currently only stabilizes tilt experienced by the PiCam. In the future, this software could further improve error by expanding to be a PID controller.

Figure 2. PID Schematic

In the data logging software, the Arduino reads in the BNO055 Euler angles and provides these values to the Raspberry Pi using serial communication. Once a BNO055 is calibrated, it provides temperature, acceleration, and position data that is useful for small-scale live-streaming applications and translates well to industrial surveillance application [6]. The serial communication between the two microcontrollers takes place via a USB connection and transfers information one bit at a time [7]. The Pi checks if the data is coming from the correct device and flushes out any I/O (input/output) buffer to avoid logging incomplete data. The available data is then decoded and written to a new .csv file every 20 seconds to minimize data loss due to possible failure modes. To accomplish this data segmentation and logging task, the python program depended on various libraries. More specifically, the os library was used to perform command line operations via python, csv was used to generate and parse .csv files, pyserial communicated with the Arduino, and datetime associated each file with its respective time stamp.

Quite often, the payloads lose power when in flight. This could mean there are chances that the data could be lost. To minimize the data lost, it is written to directories. The Pi is programmed such that it would make a new directory each time the pi restarts. With power on, the program will stay in the loop and continue writing data to CSV files in one particular directory. When the power goes off and comes back on, the program runs from the top, makes a new directory and starts writing data to newly created CSV files in that directory and the cycle continues. Like the CSV files, the directories are numbered to avoid rewriting of data to the same directory. This way of programming also helps to ensure the data lost is only for those seconds when the payload will be flying with no power. During the last flight, LIVE seemed to have lost data of at most 20 seconds.

In order to increase reliability and modularity of the system, each component of the flight software suite was built into a Raspberry Pi service to be automatically activated at system start up [8]. These services are vital to the successful operation of our software, due to the fact that the system does not have a monitor or any other peripherals that would allow us to manually activate the necessary scripts. Having the software automatically execute when the system boots up prevents any situation where the system would cease to log data or record and stream video following a loss of power or system reboot. This configuration also makes launch preparation and launch of the system more efficient and convenient, as the entire system can be powered on and flight-ready via the flip of a switch. Also, due to the fact that services are packaged applications, they can be easily and modularly added to other software suites, making them ideal for use in any case where software is needed to be run at system start up.

For a high altitude live-streaming system, incorporating a failsafe to mitigate any data loss or corruption is critical. The LIVE payload software was written in order to ensure that any possible data loss would be minimal by segmenting the recorded video and logged data and saving it as individual files throughout the duration of the flight. As a result, the post-flight analysis software suite contains data and video merging scripts to process the data and video into one compiled data and video file, respectively. Furthermore, the game engine Unity was used to visualize the BNO055 Euler angles and further visualize the rotational forces experienced by the payload during flight [9]. The Unity program utilizes multiple C# scripts to create an effective and representative simulation of the payload's orientation. This program relies on a CSV file input, containing the X, Y, and Z Euler angles of the payload during the flight. The game engine takes a CAD model of the payload and manipulates the Euler angles of this object to model the orientation of the payload during flight. Utilization of this Unity program to visualize real flight data will be discussed in the "Testing and Results" section of this paper.

V. Payload Structural Considerations

The structural system of LIVE (see fig. 1) serves two primary functions: physical protection and natural stability. It consists of two interlocking "spine" pieces which serve as the central internal surface on which most electronics are mounted. All electronics were locked in place either through heat set inserts or custom mounting in order to prevent any disconnections, damages, or unaccounted for dynamics during the flight. Around the spine pieces there is a cylindrical LEXAN shell and four carbon rods placed equidistant between each spine wall, all of which is held in place by two "caps," completing the overall cylindrical design. The structure is connected to the balloon via the payload string, which attaches to a swivel and then splits four ways. Each of these four substrings feeds through the top cap and ties onto an eyebolt set in the bottom cap.

The primary concern for the structural design was physical protection of the internal components. Since this would be the first LIVE design to fly, a high safety factor was decided upon to be sure that the onus was on the software for the functioning of the overall system. To offset any increases in weight which would come from increased safety measures the structure was also made as small as possible, limited by the battery and nanostation dimensions. The spine pieces were 3D printed from PETG plastic, and multiple thicknesses and infills were tested in order to ensure a stable and durable frame for the electronics to be mounted on. For additional support in the gaps between the spine pieces we placed carbon rods close to the outer diameter, chosen due to their high strength relative to weight. Around all of this we placed a LEXAN cylinder, having chosen LEXAN due to its moldability, high strength, and opaqueness (since the PiCam would have to be able to record video through it). The two caps were made from the same 3D printed material as the spine pieces, and each cap was attached to the spine via two steel u-bolts. The caps were also designed to geometrically lock both the LEXAN and the carbon rods in place once the u-bolts were tightened. As a final safety measure, two worm clamps were tightened around the LEXAN exterior.

The secondary concern was to design a structure which would minimize the natural motion of the payload during the launch in an effort to give our proportional controller system more ideal conditions to work with. The overall cylindrical shape of LIVE was developed so there would be no uneven aerodynamic surfaces which might cause the payload to begin spinning uncontrollably in wind [10]. As an additional measure, we occupied the bottom spot on the payload string and included a swivel just above LIVE to negate the string itself from spinning due to

atmospheric forces on payloads positioned higher up. In order to minimize the potential for swinging, we designed LIVE to be bottom-heavy. Additionally we placed the eye bolts in the bottom cap as close to the outer diameter as possible, creating a wide base for tying on and increasing hanging stability.

Figure 3. Fully Assembled Payload

VI.Testing and Results

The LIVE payload system was flown on a tandem high-altitude balloon launch. This flight consisted of our payload at the bottom and another payload at the top. Our payload has to sit at the bottom of the payload train to ensure a solid wireless connection to the ground station. This flight was a partial success for the team, as we weren't fully able to capture and stream to the internet as the payload was designed for but we got some good overall data and locally saved video from our flight. This data was key in understanding the successes and failures of our payload's first iteration.

The payload's software suite was successfully able to record video and log the BNO055 data from the flight. The video and data segmentation measures incorporated into the suite operated as intended, and the segmented files were successfully merged together with our post-flight analysis software suite. Prior to the launch of the balloon, a successful live-stream connection was established with the payload system. However, shortly after the launch, this connection was lost and unable to be re-established due to the ground station antenna hardware being out of range of the Nanostation signal. Although connection to the live-stream of the test flight was lost, post-flight analysis of the merged video clips confirms that the payload hardware and software were successful in capturing the video feed. Also, since an average balloon flight of no longer than 2.5 hours was expected, the video recording software was written to only capture video for this length of time. Due to unforeseen delays to the launch of the balloon, the longer flight time as a result of the tandem balloon configuration, and unaccounted time to power on and prepare the payload, LIVE was unable to capture video from the entire flight. For future flights, the video capturing time will be increased in order to take these time delays into account. The BNO055 data was logged as intended, but upon closer inspection, some of the lines of the csv output file contained uncalibrated data entries. The high turbulence that the payload was subjected to during its flight is likely responsible for these periods of lost calibration.

The hardware of the payload functioned as expected. While leaving the launchpad, LIVE was streaming locally. As expected, as the payload left our line of sight, we lost connection until the payload was about 10 km in the atmosphere. This loss of connection is due to a lower end ground station antenna choice the team went with. This choice works well with the payload configuration but can be upgraded for increased range. However, due to high-speed winds, our payload traveled in a different direction than we expected in tracking. Our video and accelerometer data, combined with data from another payload showed that one of our balloons burst at approximately 18km during flight due to a faulty balloon neck insert. The balloon wasn't high enough nor close enough to stream video during flight due to the range of our ground station antenna. The video and data were saved successfully. Due to our payload's exterior housing, the payload internal temperature reached a maximum of 49 ℃. This was achieved through software optimization in the Raspberry Pi. We also had two separate impact events both before launch and during landing. The electronics showed no damage and functioned properly according to post-flight diagnostics. All wires were intact and the systems did not show any signs of wear. The battery also seemed to have a lot of life left so we have decided to go with a smaller size Li-Po for the next launch to reduce weight.

In order to enhance our post-flight data analysis, a Unity game engine program was created to provide a visual representation of the BNO055 Euler Angle data. The game engine takes a CAD model of the payload and manipulates the Euler angles of this object to model the orientation of the payload during flight. Paired with the video footage from the payload itself, the Unity simulation allows us to develop a better understanding of the flight conditions, and determine the true effectiveness of our video stabilization efforts via the proportional controller software.

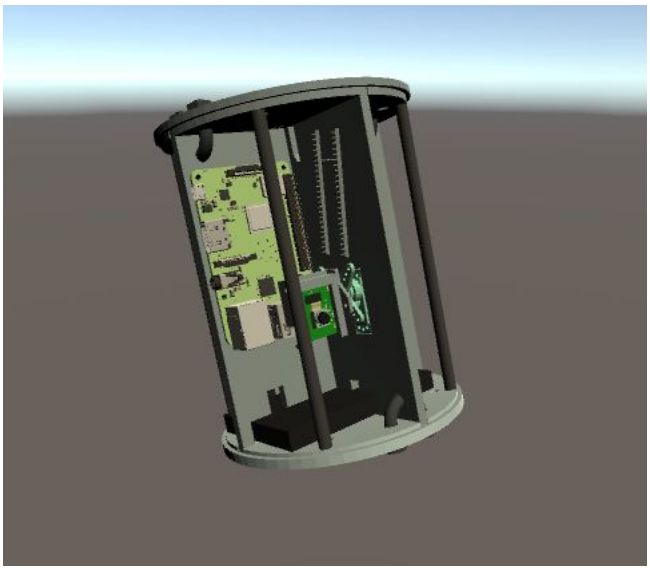

Figure 4. Unity Animation Using BNO055 Data

Acknowledgments

The authors of this paper would like to thank the University of Maryland's Balloon Payload Program with their help in developing and testing our payload, LIVE.

References

[1] Smith, M., & Rainwater, L. (2003). Applications of Scientific Ballooning Technology to High Altitude Airships. *AIAA's 3rd Annual Aviation Technology, Integration, and Operations (ATIO) Forum*. doi:10.2514/6.2003-6711

[2] Sushko, A., Tedjarati, A., Creus-Costa, J., Maldonado, S., Marshland, K., & Pavone, M. (2017). Low cost, high endurance, altitude-controlled latex balloon for near-space research (ValBal). 2017 IEEE Aerospace Conference. doi:10.1109/aero.2017.7943912

[3] Shi, C., & Bhargava, B. (n.d.). An efficient MPEG video encryption algorithm. Proceedings Seventeenth IEEE Symposium on Reliable Distributed Systems (Cat. No.98CB36281). doi:10.1109/reldis.1998.740527

[4] University of Maryland Balloon Payload Program, "UMDBPP/LIVE", https://github.com/UMDBPP/LIVE

[5] N. Passalis, A. Tefas and I. Pitas, "Efficient Camera Control using 2D Visual Information for Unmanned Aerial Vehicle-based Cinematography," 2018 IEEE International Symposium on Circuits and Systems (ISCAS), Florence, 2018, pp. 1-5.

[6] Bosch Sensortec, "BNO055 Intelligent 9-Axis Absolute Orientation Sensor", https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bno055-ds000.pdf

[7] Pomarlan, Mihai. *Evaluating the usefulness of USB for realtime robotics applications*. n.d. <http://www.etc.upt.ro/documents/studenti/proiecte_diploma/pomarlan.pdf>.

[8] Parnas, D. L., Clements, P. C., & Weiss, D. M. "The modular structure of complex systems," IEEE Transactions on Software Engineering, 11(3), 259-266. (1985).

[9] C. Donalek *et al*., "Immersive and collaborative data visualization using virtual reality platforms," *2014 IEEE International Conference on Big Data (Big Data)*, Washington, DC, 2014, pp. 609-614, doi: 10.1109/BigData.2014.7004282.

[10] NASA Drag of a Sphere. Retrieved September 15, 2020, from https://www.grc.nasa.gov/www/k-12/airplane/dragsphere.html