SPARC: Space Particle and Radiation Counter

Smith Barrionuevo, Mackenzie Olson, Shreya Santosh, Laura Tisher

University of Colorado at Boulder

NASA Colorado Space Grant Consortium DemoSat

4/13/2026

Table of Contents:

Abstract:

The objective of our payload is to find numerical correlations between solar radiation in space and the energy that solar panels output. This research is important in the context of spacecraft power subsystems, to understand and better develop solar systems that take full advantage of radiation in space. By doing so, cost can be brought down, and power efficiency can be brought up, overall making a spacecraft much more effective. The data collected by our various solar panels and sensors that read different radiation spectrums describes the magnitude of radiation. The magnitude of intensity of UV channels is ranked by UVB, UVA, and then UVC. Furthermore, our visible spectrum is measured by our visible light sensors, which spike when exposed to bright light. Our near-IR channel works similarly but provides more detailed data than the Visible spectrum. The temperature can vary heavily, especially at an altitude of 100,000 ft. It is critical to track the temperature of our payload to ensure all our components are warm enough to function efficiently. Our instrumentation has been able to accurately provide data on solar panel voltage generated when exposed to certain brightness and radiation. The team hypothesizes that solar panel voltage output will positively correlate with measured radiation intensity (UV, VIS, IR) and increase with altitude due to reduced atmospheric attenuation.

Materials and Methods:

Science

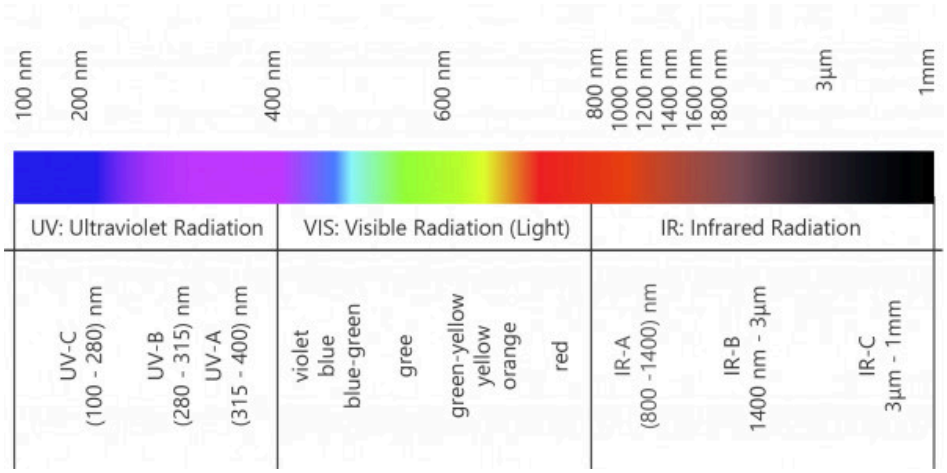

Figure 1.1

Based on radiation levels at around 100,000 ft, we specifically chose UV, IR, and Visible light to collect data on the most frequent and relevant radiation spectrums in the context of solar panel readings. The biggest science concepts and theories that we will be testing include wavelengths in the electromagnetic spectrum and how they interact with our sensors. A lot of research has been put into studying the Sun and its radiation. Currently, the GOES X-ray flux from NOAA measures the real-time radiation emitted by the Sun. There is also a Spacecraft named SWPC in L1, which tracks real-time solar winds and is one of our safety measures for warning against dangerous solar activity, with a half-hour warning. These scientific measurements, among others, are used to better develop our society as a whole. Urban planning, weather forecasting, and understanding space and radiation overall. For our project, we specifically want to observe how larger radiation doses might result in higher solar panel efficiency.
The sensors on the payload should track the following parts of the light spectrum:

Spectral Sensor - UV:

- UVC (200-280nm)
- UVB (280-320nm)
- UVA (320-400nm)

Spectral Sensor - Visible Light (VIS): (including 40nm of full-width half-max detection)

- Violet (450nm)
- Blue (500nm)

- Green (550nm)
- Yellow (570nm)
- Orange (600nm)
- Red (650nm)

Spectral Sensor - Near Infrared: (including 20nm of full-width half-max detection)

- R2 (610nm)
- S (680nm)
- T (730nm)
- U (760nm)
- V2 (810nm)
- W (860nm)

Hardware

The hardware for this project provided all data collection for both the DEMOSAT payload and the ground system. This hardware consisted of the following components:

- 3x Arduino Uno
- 5x Solar Panel (5V)
- 5x Voltage Divider
- 3x OpenLog
- 2x LED
- 2x Spectral Sensor - Near Infrared
- 2x Spectral Sensor - Visible Light
- 2x Spectral Sensor - UV
- 1x Altitude/Pressure Sensor
- 3x Battery (9V)

Multiple rounds of stringent testing followed the acquisition of all hardware to guarantee effectiveness in testing conditions. Additionally, the power system was tested thoroughly to guarantee power delivery to all components. It was during this testing with the payload system that we discovered insufficient power delivery to the OpenLog SD card reader, which was causing critical failure in data logging. To rectify this, the payload system was split into two separate Arduinos, code edited, and power delivery systems split onto two separate batteries (Fig. 2.2 and 2.3). The ground system retained the original design (See Fig. 2.1).

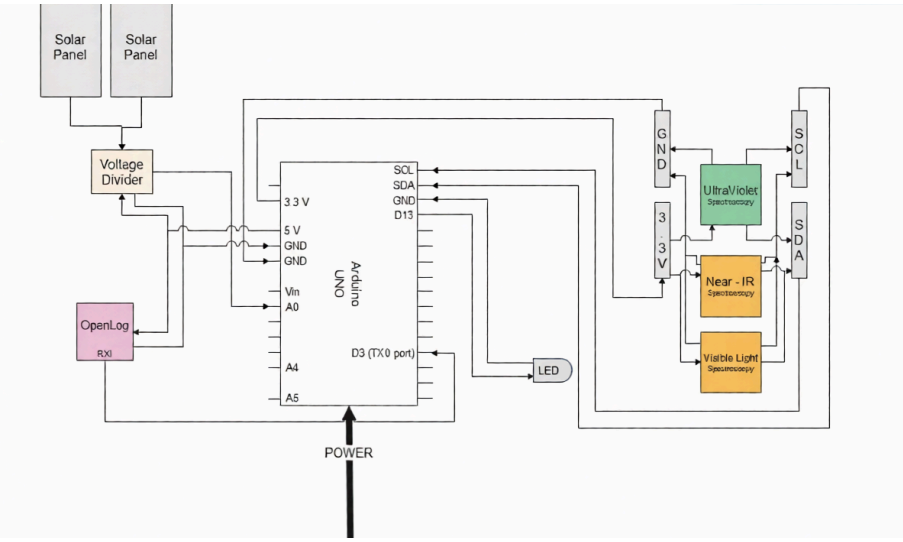

Figure 2.1: Ground System Electronics Schematic

Note: (The ground system had one less solar panel than the payload system)

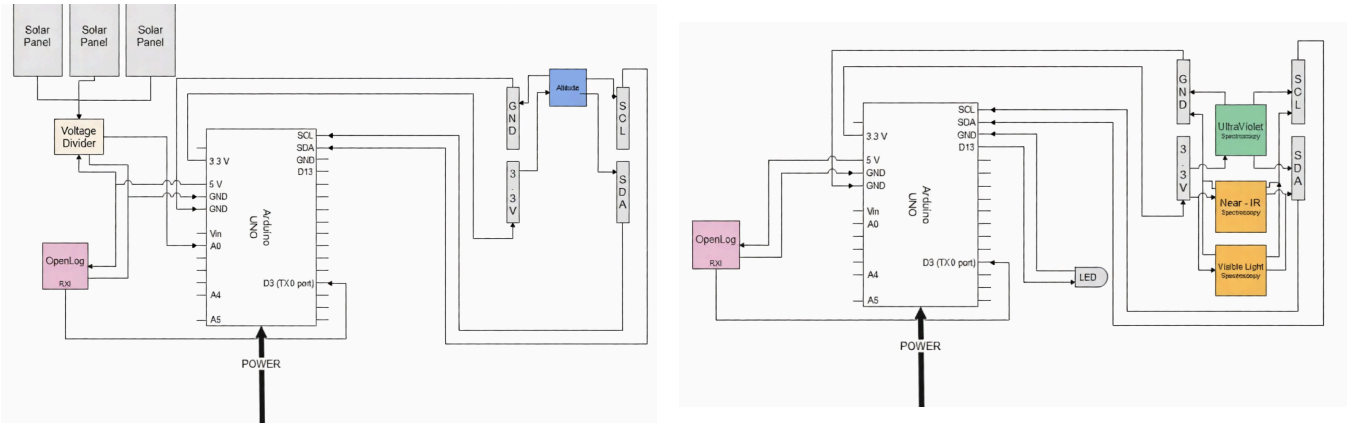

Figure 2.2 and 2.3, Solar and Altitude Schematic (Left) and Spectroscopy Schematic (Right):

Both payloads used a 9V battery for power, connected to basic switch circuits for power control.

Note: (Only the spectroscopy Arduino had a confirmation LED)

Software

The software for this project consists of microcontroller firmware, data logging systems, and data processing as its main components. In terms of firmware, the majority of coding was done in C++ on Arduino's IDE for a seamless connection between the code and the Arduino. Arduino's IDE lends itself to streamlined sensor implementation by allowing easy access to downloadable libraries that are offered for sensors, such as the spectral sensors on our payload (see Appendix A.1 for code).

For sensor implementation, careful configuration of operating parameters was necessary to ensure reliable measurements under expected timing and environmental conditions. Both the UV and spectral light sensors had adjustable gain settings, which were selected after a few rounds of testing under the anticipated launch conditions (moderate to direct sunlight). Decreasing the gain settings ensured that the sensors would have an appropriate level of sensitivity for high light input, while maintaining sufficient range to account for cloud cover. These parameters were configured during system initialization before the start of the data acquisition cycle. If any sensor failed to initialize, the system entered an error state indicated by a continuously blinking external LED. Otherwise, successful initialization was confirmed by three short LED flashes, which provided a simple visual verification of system readiness before launch.

Data acquisition was performed within the main program loop of the microcontroller. During each acquisition cycle, all the sensors would read data in a sequential order before a brief delay to allow for sensor integration times and to ensure that measurements were fully completed before data retrieval. Following this delay would be when a line of the data is written into the SD card with 3-5 places of decimal precision depending on which type of data was being taken in (see sample data in Appendix C.1). The data was formatted such that it could be converted to a CSV file, which would be useful for data processing and analysis. To improve logging reliability, several software safeguards were implemented, including a multi-second delay after serial initialization to allow the OpenLog module to fully boot and mount the SD card and a first-write guard to ensure that the file header was written only after the system reached a stable operating state. These measures mitigated previously observed issues with corrupted file headers and incomplete writes during startup.

The analysis step was handled in MATLAB, in which scripts were written to import the recorded CSV data using table formatting. Then, plots were generated for all of the different data that was collected from the payload and on the ground, split between using time and altitude as the independent axes. With the sheer amount of data there was to compare, it was necessary to have many plots to analyze and draw conclusions from (see Appendix B.1 for plot script example). MATLAB also proved to be useful in handling flawed data. Some data that was collected during the testing phases, as well as during the flight itself, contained corrupted ASCII characters instead of coherent data (see Appendix C.2 for corrupted sample data). Analysis of these corrupted files revealed that the underlying data remained intact but had been misinterpreted due to byte inconsistencies caused by the OpenLog. Because the errors followed a consistent pattern, MATLAB scripts were able to be used to reconstruct the data by correcting the byte alignment, thus providing a way to recover all data successfully. An abridged version of the recovery code can be found in Appendix B.2.

Structure:

The main structure of the payload is made out of black foam board, aluminum tape, and hot glue. The structure was designed to be the smallest size box, while having a large surface area on the top to fit all of our sensors. The final box was designed to be 16x16x12 cm. Please see the CAD model (Fig. 3.1) below for more details.

The top of the box holds all the sensors in place and protects them during landing. From the top view, you can see 3 red squares, the UV, near IR, and visible light sensors, 3 solar panels, and the flight straw integration.

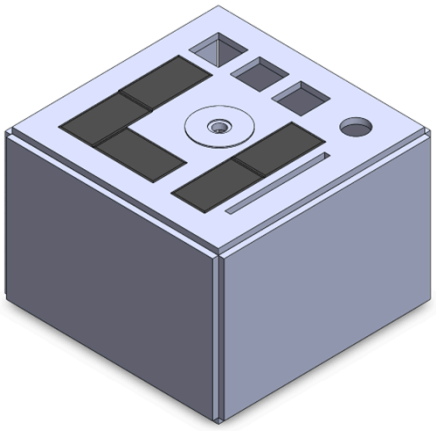

Figure 3.1: Initial CAD Model of Payload

Note: (Final design had two fewer solar panels)

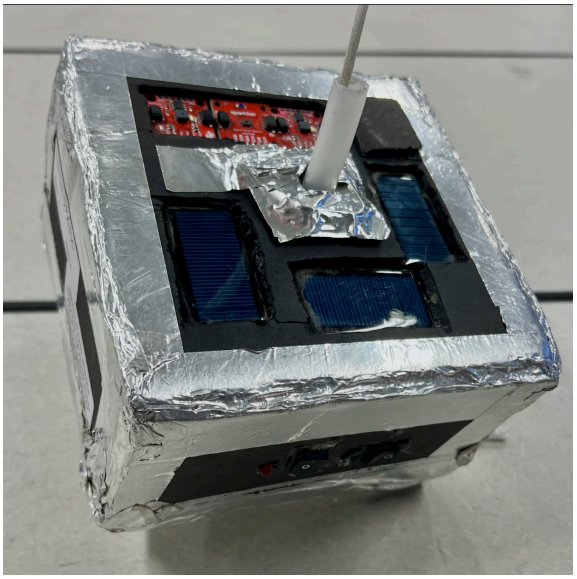

Figure 3.2: Final Payload After Flight

Flight Straw Integration:

The flight straw allows a string to be run through the payload to be connected to the balloon. The flight straw is a 1cm diameter plastic tube that was inserted from the top to the bottom of the payload. 1cm diameter holes were cut through the black foam core at the center of the top and bottom panels of the box. Hot-glued plastic washers on the outside of these cutouts ensured the flight straw couldn't cut through the foam core during flight. On flight day, an exact knife was used to poke small holes in the flight tube where it would intersect the top and bottom of the payload structure. A bent paper clip was fed through this hole and bent flat on both sides of the plastic washer to secure the flight straw to the top or bottom of the payload. Then, small strips of aluminum tape were used to secure the paperclip down to the washer. This ensured the flight straw was fully and strongly integrated into the payload(See Fig. 3.2).

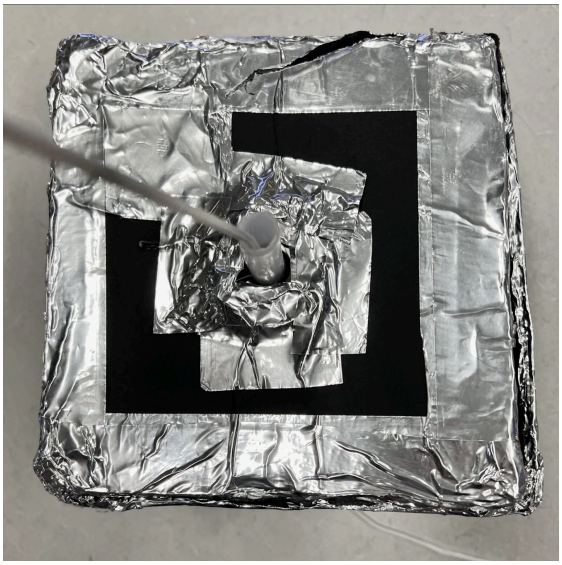

Figure 3.3: Flight Straw Integration

Thermal System:

One concern of the payload reaching space was the cold temperature it would experience. To ensure the payload would stay warm, a thermal system was created to trap heat inside the system. Black 1 cm thick foam insulation material was cut into sections to fit the dimensions of the box. Each piece was cut 1.5cm shorter on all edges to fit while the payload was taped together. For the top of the box, the ones with sensors attached and wires exiting this side, smaller pieces of the insulation were cut to place around the wires. The insulation helped to trap heat inside the payload.

Testing proved that the insulation was not enough alone to ensure the payload stayed warm, so air-activated hot hands were also included in the payload. Testing proved that 3 hot hands were sufficient to keep the payload at an operating temperature for a 3-hour period. See the testing section for more details.

Testing and Flight Readiness:

Structural Testing:

Many tests were completed to ensure that the payload was flight-ready. These tests were composed of two sections: structural testing and functional testing.

For structural testing, the four tests listed in the DemoSat user guide were performed. These included the drop test, stair pitch test, shake test, and whip test. Please see the diagram below for initial results based on the first round of testing.

Test	Date Completed	Outcome
Drop Test	3/30/2026	Fail
Stair Pitch Test	3/30/2026	Pass
Shake Test	3/30/2026	Pass
Whip Test	Early March	Fail

After the first round of testing, two tests were not passed: the drop test and the whip test. The drop test involves dropping the payload from the second story of a building and observing whether the internal components survive. Although not all the hardware was integrated, components that were inside the payload broke and shifted around.

The whip test was also unsuccessful. This test involves attaching a string through the flight straw and whipping it around to ensure that the payload is securely integrated with it. The payload was not securely attached to the structure and needed to be remedied before launch. To fix this, hot glue was used to secure plastic washers around the hole for the flight straw, and paper clips were added through the sides of the flight straw, bending them to reinforce the connection. After retesting, we confirmed that the flight straw was securely attached to the payload. See the graphic below for more specific details on the tests

performed and the changes made.

Test	Details
Drop Test	LED light went off, concluded SD card broke. We found a new SD card and have velcroed the SD card reader inside. Camera broke, decided to not use camera.
Stair Pitch Test	All components worked and data was recording.
Shake Test	Added more foam around components inside to ensure they were supported and locked in place.
Whip Test	Reglued flight straw plastic washers and added paper clips to secure flight straw to payload

After the second round of structural testing, the payload passed the whip test, shake test, and stair pitch test. However, the drop test failed again. The main failure occurred with the small camera mounted in the upper corner of the box. It was not secured properly and fell during the drop, ultimately shattering at the bottom of the payload. Due to time constraints, the team decided to remove the camera and seal the opening with additional foam board and hot glue. Although it was disappointing not to collect photos or videos during the flight, this was determined to be the best decision to ensure overall payload success.

Functional Testing

Several functional tests were completed to ensure the payload operated properly throughout its flight and under expected environmental conditions. These included the cold test, sensor test, and full functional testing.

An initial cold test was conducted at the beginning of March to determine whether the small heaters we purchased would maintain sufficient temperature inside the payload. From this test, it was discovered that the heaters drained power too quickly and could not maintain temperature for more than 10 minutes, meaning they would not last for the full flight duration of nearly three hours.

On the same day, we conducted an initial sensor test, which ran for 10 minutes and collected data every few seconds. The sensors performed well and recorded reliable data. A few weeks later, on March 15, we completed an updated sensor function test lasting 15 minutes. During this test, a timing function was added to improve sensor performance, resulting in clearer and more useful data for analysis.

At the end of March, a full functional test lasted approximately three hours. While data was collected, the payload lost power after about 2.5 hours.

After making adjustments, another cold test was conducted. This test lasted 30 minutes, as the payload at that time could only operate for that duration. As a result, only 30 minutes of usable data were obtained. Please see the table below for specific testing details.

Test	Date for Test	Duration of Test	Notes
Initial Cold Test	3/2/2026	10 minutes	Heaters drained power too quickly
Initial Sensor Test	3/2/2026	10 minutes	Sensors respond and record data well
Updated Sensor Function Test	3/15/2026	15 minutes	Added time function and improved sensing
Functional Test	3/26/2026	~3 hours	Reasonable data, died after 2.5hrs
Cold test	3/30/2026	30 minutes	Died after 30 minutes, had usable data for 30 minutes.

Following these unsuccessful cold and functional tests, the main issue was identified: the payload's power supply. To address this, the systems were separated; the solar panels and altitude sensor were powered by one 9V battery, while the spectroscopy sensors were powered by a second 9V battery. Each system also had its own OpenLog and SD card, allowing us to collect data independently. This required duplicating switches and components, which increased the payload weight, but still remained under the target launch weight.

With these changes, both the functional and cold tests were completed successfully. On flight day, our payload operated for approximately 1.7 hours, which matched the shortened flight time due to weather conditions that caused the balloon to be popped early. Data was collected for most of the flight, providing a satisfying outcome.

Results and Analysis:

It may be useful to reference the following information when viewing the plots: The ultraviolet part of the spectrum is broken down into 3 wavelengths: UVA, UVB, and UVC. Similarly, the visible light spectrum is broken into the range of colors (R being red, O being orange, etc.), and the IR part of the spectrum is broken into 6 wavelengths (ranging from R2 to W). Temp1 and Temp2 refer to the temperatures collected on each of the spectroscopy sensors (which were exposed to the air outside of the payload, as well as *some* warmth from inside the payload), while TempAlt refers to the temperature collected by the altitude sensor (which was *inside* the payload and exposed to the full heating). Any mention of Temp on its own is a mean of these temperatures.

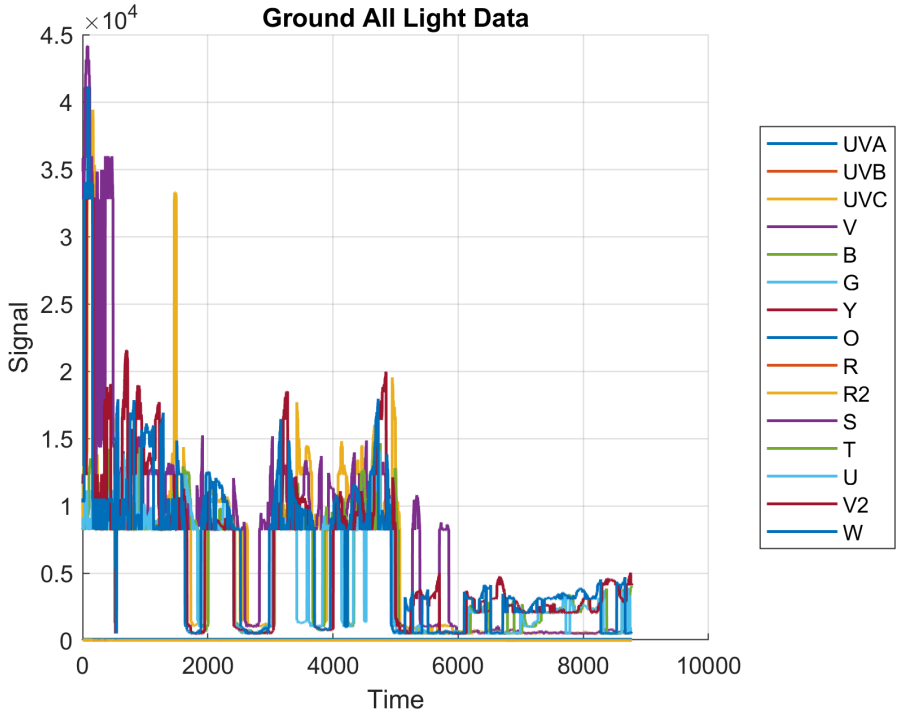

Figure 4.1: All Light Data for Ground System - Ground System's detection of different radiation sensors. Most points make sense, as radiation drops after a certain point and stays within a relative range of each other. The drop in overall light data can be linked to payload descending, and therefore having less light and overall radiation. Violet spikes the most overall, while UVA is the first radiation to drop over time. The drop is noted at around 5000 seconds.

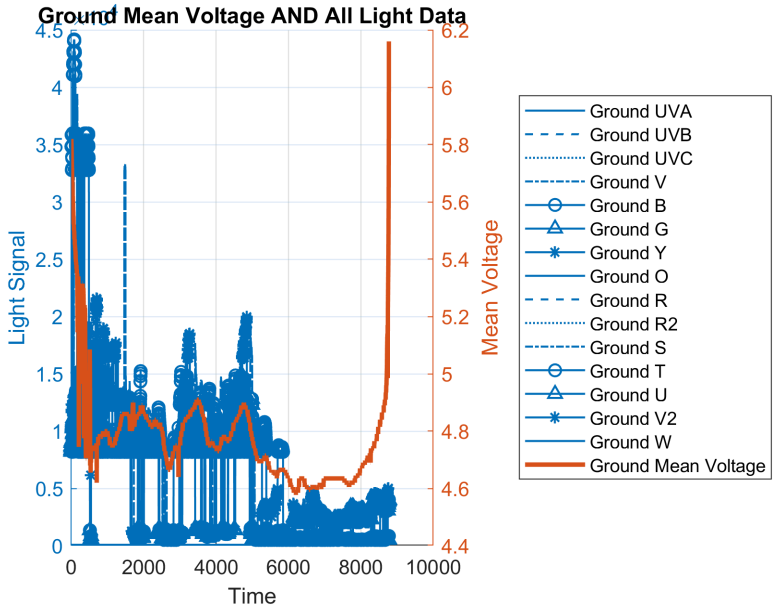

Figure 4.2: Average Voltage from the ground system compared to all radiation data over experimentation time. The average voltage varies depending on the light data, with a correlation of higher light resulting in higher voltage detected. This helps back up our research that more radiation results in more voltage generated overall. There is a sharp spike at the end of our data set, which is likely linked to the ground system being turned off.

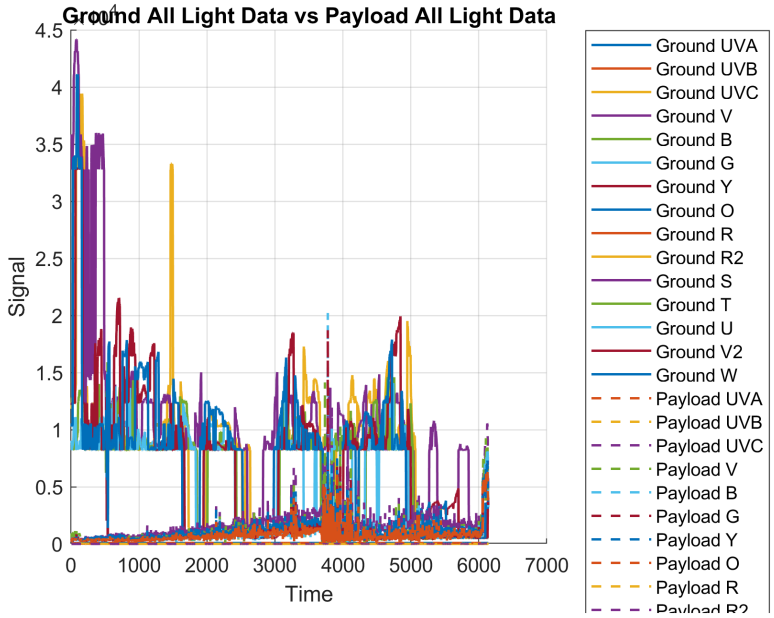

Figure 4.3: Radiation data from the ground system and payload. Ground data is at a higher baseline than payload data. Again, we see Violet being an outlier by having consistently high signals in terms of radiation. There is a baseline discrepancy, which suggests that our pair of radiation sensors was on

different sensitivities and scales. However, they have the same baseline, which helps us compare the data effectively.

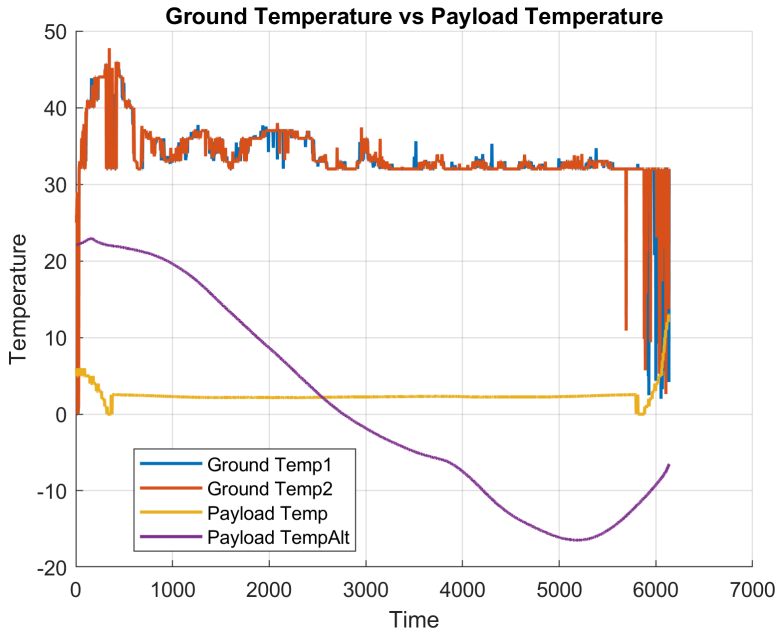

Figure 4.4: Ground Temperature vs Payload Temperature. Payload temperature drops steadily as it gains altitude. The curve goes upwards as the payload becomes warmer, due to the descent. Furthermore, from the ground temperature, we can see when the Sun rose and therefore caused a spike in temperature from the sensors. This spike gradually lessens, indicating the temperature is stabilizing. The temperature has an abnormality at the end, again suggesting issues with turning off the system.

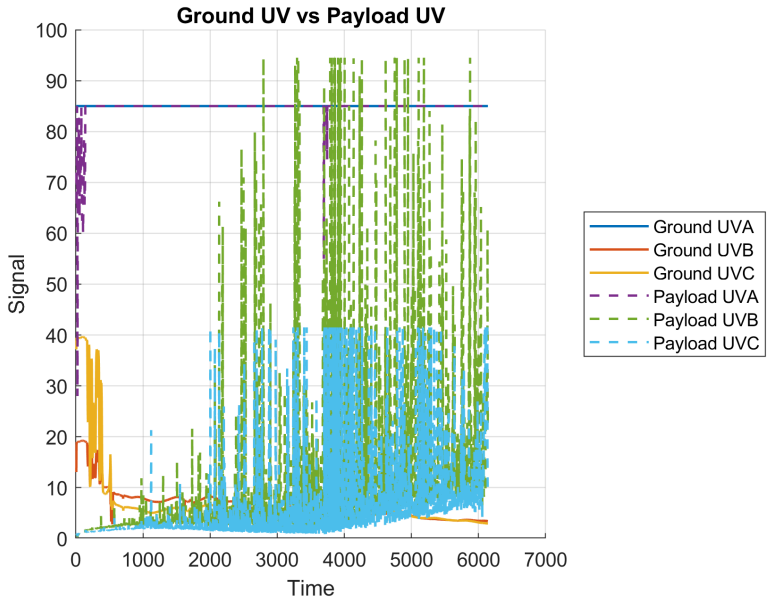

Figure 4.5: Ground UV vs Payload UV. Higher UV radiation levels are found in the payload, which follows our research hypothesis. Payload UVB spikes consistently, which helps us conclude that it was the most prominent radiation detected on the day of launch. Payload UVC is also prominent on the graph, but it seems to falter at half the magnitude of Payload UVB. Initially, Payload UVA and Ground UVC spikes intensely before flattening out.

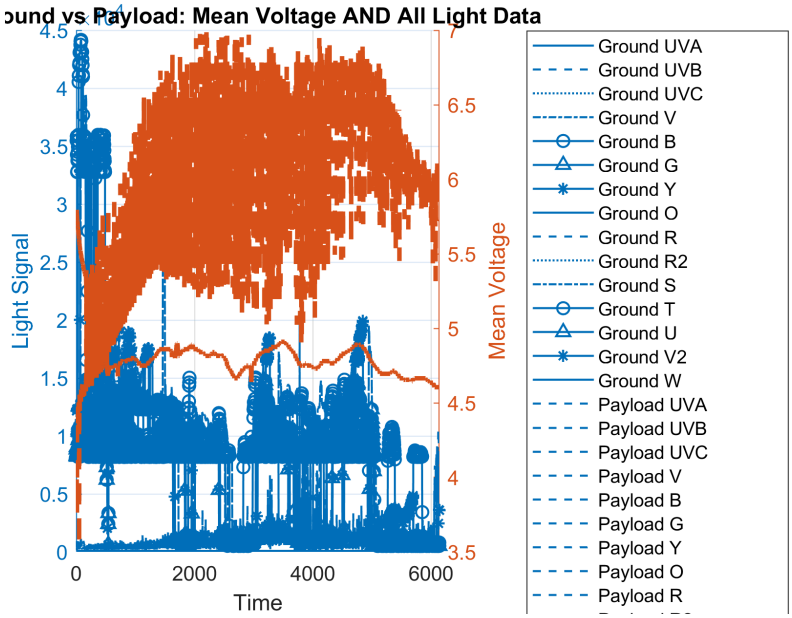

Figure 4.6: Average Voltage in correlation with all light data found on the ground and payload system. Voltage spikes at higher visible lights. The orange means voltage line spikes and dips in unison with light and radiation. Voltage efficiency peaks right before the 4000 time stamp. This point marks the height of the payload at around 80,000 ft, where light and radiation are at their highest point, resulting in the best solar panel efficiency.

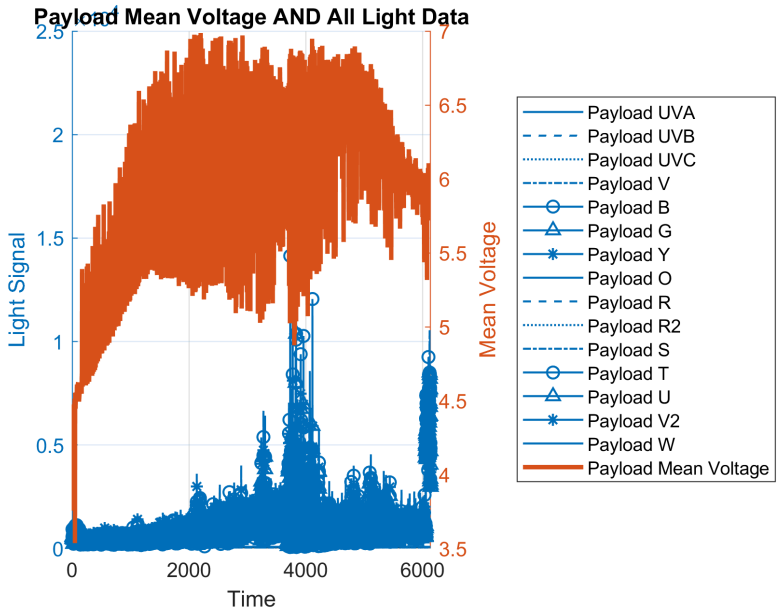

Figure 4.7: Average voltage on payload compared to all light data, including radiation. The average voltage of the graph increases steadily as the payload ascends in the atmosphere, and voltage spikes in the ozone layer due to more UV radiation. Around the 4000 time stamp mark, we see light spikes and the voltage slightly drops. This point is a great indicator of where the balloon has popped. We then see the voltage dropping as the payload also drops down to Earth.

Conclusion and Recommendations:

SPARC successfully collected visible (VIS), UV, and IR radiation, which illustrated meaningful data between radiation exposure and solar panel efficiency. The average voltage of the payload spiked exponentially during ascent, which peaked around 7V before stabilizing to 5.5V - 6V. This dataset suggests that our scientific hypothesis was correct, as a decrease in atmosphere and an increase in solar irradiance improved voltage output. UVA has the strongest individual correlation between detected wavelength and voltage. Temperature is also a dominant factor for voltage variation in the context of more efficiency, as cooler cells perform better at lower temperatures. UVB and UVC increased drastically at time step 2,500s, which is consistent with passing through the ozone layer (which typically blocks most UV radiation). The external temperature of the payload dropped to a low of -16°C; Our thermal subsystem was a success, maintaining an internal temperature of about 2-3 °C, which was a great working baseline for all radiation sensors onboard. The ground system recorded higher VIS than our payload, which may signify different orientations for sensor magnitudes, and or shading conditions.

Overall, our data support our research that radiation exposure increases solar panel voltage output, including other factors such as a thinner atmosphere and lower temperatures. Some techniques to implement for similar projects in the future would be ensuring there is redundant voltage logging, as there is some voltage instability noted late in the flight. Furthermore, using more specialized solar radiation detectors can help narrow and specialize the data collected.

Our collected data and synthesized research will help develop solar systems that take full advantage of radiation in space, optimizing solar panels for wavelengths not normally found on Earth. By doing so, cost can be decreased, and power efficiency can be increased, allowing for higher voltage usage by satellites and increased payload viability.

References

Lincot, D., & Ibarrart, L. (2026, March 4). *Ultra-lightweight, high-performance, foldable: Advances in space photovoltaics*. Polytechnique Insights. https://www.polytechnique-insights.com/en/columns/space/ultra-lightweight-high-performance-foldable-advances-in-space-photovoltaics/

Solar performance and efficiency. (n.d.). Energy.gov. https://www.energy.gov/cmei/systems/solar-performance-and-efficiency#:~:text=When%20an%20electron%20encounters%20a,than%2030%25%20of%20incident%20light.%E2%80%8B

The optical radiation wavelength range. (n.d.). Gigahertz-Optik. https://www.gigahertz-optik.com/en-us/service-and-support/knowledge-base/basics-light-measurement/light-color/opt-rad-wavelength-range/

Appendix

Appendix A - Microcontroller Firmware

A.1 - Ground System Code

The following listing contains the complete ground system Arduino firmware used for sensor initialization, data acquisition, and logging to the OpenLog module. The ground system code is representative of what is in both of the separate code files used for the payload.

```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h> // Include the SoftwareSerial library

// OPENLOG SETUP
// Connect OpenLog TX to Arduino Pin 2 (RX)
// Connect OpenLog RX to Arduino Pin 3 (TX)
SoftwareSerial openLog(2, 3);

//UV
#include <SparkFun_AS7331.h>
SfeAS7331ArdI2C myUVSensor;

//Altitude
#include <SparkFunMPL3115A2.h>
MPL3115A2 myAltitudeSensor;

//Spectral
#include <AS726X.h>
AS726X sensor; // Creates the spectral sensor object
byte GAIN = 2; // 0 = 1x, 1 = 3.7x, 2 = 16x, 3 = 64x gain
byte MEASUREMENT_MODE = 2;  // Continuous reading of all channels

//Voltage
float volt1;
float volt2;
float volt3;

const int STATUS_LED = 13; // Define the pin for the LED

// Function that flashes the LED forever if something breaks
void errorFlash() {
```

```
  while (true) {
    digitalWrite(STATUS_LED, HIGH);
    delay(200); // Fast flash on
    digitalWrite(STATUS_LED, LOW);
    delay(200); // Fast flash off
  }
}

void successBlink() {
  for (int i = 0; i < 3; i++) {
    digitalWrite(STATUS_LED, HIGH);
    delay(200);
    digitalWrite(STATUS_LED, LOW);
    delay(200);
  }
}

void setup() {
  Serial.begin(115200);
  // Set the baud rate (default baud rate for most SparkFun OpenLogs is 9600)
  openLog.begin(9600);

  // Set the LED pin as an output
  pinMode(STATUS_LED, OUTPUT);

  while (!Serial) {
    delay(100);
  };

  while (!Serial) { delay(100); }
  Serial.println("Starting up sensors...");

  // Initialize I2C bus
  Wire.begin();

  // UV SENSOR SETUP (AS7331)
  if (myUVSensor.begin() == false) {
    Serial.println("UV Sensor failed to begin. Please check your wiring!");
    errorFlash();
  }
  Serial.println("UV Sensor found and connected.");
```

```
  // Set gain to 512 (Default is 2, Max is 2048)
  myUVSensor.setGain(GAIN_512);

  // Increase conversion time to 512 milliseconds to collect more light
  myUVSensor.setConversionTime(TIME_512MS);

  // Set device operating mode to Command/One-Shot mode
  if (myUVSensor.prepareMeasurement(MEAS_MODE_CMD) == false) {
    Serial.println("UV Sensor did not set properly.");
    errorFlash();
  }

  // ALTITUDE SENSOR SETUP (MPL3115A2) for payload
//myAltitudeSensor.begin(); //Get sensor online
//myAltitudeSensor.setModeAltimeter();   //Set to measure in meters
//myAltitudeSensor.setOversampleRate(4);
//myAltitudeSensor.enableEventFlags();   //Enable event flags for proper reading
//if (myAltitudeSensor.readAltitude() == -999)
//   {
//  Serial.println("Altitude Sensor did not set properly.");
//  errorFlash();
// }
// Serial.println("Altitude Sensor found and connected.");

  // SPECTRAL SENSOR SETUP (AS726X)
  if (sensor.begin(Wire, GAIN, MEASUREMENT_MODE) == false) {
  Serial.println("AS726X Spectral Sensor failed to begin. Please check your wiring!");
  errorFlash(); // Trigger the flashing light!
}
  Serial.println("AS726X Sensor initialized.");

  successBlink();

   //Log file header (CSV header to the SD card to name columns)

//Altitude is included in between UVC and V, and TempAlt is at the end for the payload
openLog.println("Time,UVA,UVB,UVC,V,B,G,Y,O,R,R2,S,T,U,V2,W,Volt1,Volt2,Volt3,TempSpec
");
}

void loop() {
  // AS7331 UV read
```

```
  if (ksfTkErrOk != myUVSensor.setStartState(true)) {
    Serial.println("Error starting UV reading!");
  }

  delay(2 + myUVSensor.getConversionTimeMillis());

  if (ksfTkErrOk != myUVSensor.readAllUV()) {
    Serial.println("Error reading UV.");
  }

// Altitude for payload:
//float altitude = myAltitudeSensor.readAltitude(); //[m]
//float tempAltC = myAltitudeSensor.readTemp();//[C]

// Spectral
sensor.takeMeasurements();

float seconds = millis() / 1000.0;

//Temperature
float tempSpec = sensor.getTemperature();

// Solar panel voltage
volt1 = float(analogRead(A0))*25.0/1023.0;
volt2 = float(analogRead(A1))*25.0/1023.0;
volt3 = float(analogRead(A2))*25.0/1023.0;

// OpenLog Printing Starts Here

//Time
openLog.print(seconds);
openLog.print(",");

// UV
openLog.print(myUVSensor.getUVA(), 5); openLog.print(",");
openLog.print(myUVSensor.getUVB(), 5); openLog.print(",");
openLog.print(myUVSensor.getUVC(), 5); openLog.print(",");
openLog.print(altitude, 2);
openLog.print(",");

//[This is where altitude is printed for the payload system]
```

```
// Visible readings (AS726X)
openLog.print(sensor.getCalibratedViolet(), 5); openLog.print(",");
openLog.print(sensor.getCalibratedBlue(), 5); openLog.print(",");
openLog.print(sensor.getCalibratedGreen(), 5); openLog.print(",");
openLog.print(sensor.getCalibratedYellow(), 5); openLog.print(",");
openLog.print(sensor.getCalibratedOrange(), 5); openLog.print(",");
openLog.print(sensor.getCalibratedRed(), 5); openLog.print(",");

// Near IR readings (AS726X)
openLog.print(sensor.getCalibratedR(), 5); openLog.print(",");
openLog.print(sensor.getCalibratedS(), 5); openLog.print(",");
openLog.print(sensor.getCalibratedT(), 5); openLog.print(",");
openLog.print(sensor.getCalibratedU(), 5); openLog.print(",");
openLog.print(sensor.getCalibratedV(), 5); openLog.print(",");
openLog.print(sensor.getCalibratedW(), 5); openLog.print(",");

openLog.print(volt1); openLog.print(",");
openLog.print(volt2); openLog.print(",");
openLog.print(volt3); openLog.print(",");

openLog.print(tempSpec); openLog.print(",");
openLog.println(tempAltC); //println adds the carriage return for the next row
// Wait a second before the next read
delay(1000); }
```

Appendix B - MATLAB Scripts

B.1 - Processing and Plotting Script (Abridged)

The following is a small snippet of the code utilized to import and visualize the data.

```
%% Weather Balloon Analysis Plots
clear; clc; close all;

%% SETTINGS
groundFile = "GroundSystemDataSheet.xlsx";
payloadSpectralFile = "SpectralDataSheet.xlsx";
payloadSolarFile = "SolarPanelDataSheet.xlsx";

savePlots = false;
outFolder = "balloon_analysis_plots";
```

```
% Altitude column candidates in payload spectral file
altitudeCandidates = {'Altitude','Alt','altitude','ALTITUDE','Height','height'};

if savePlots && ~exist(outFolder,'dir')
    mkdir(outFolder);
end

%%  LOAD DATA
G  = readtable(groundFile, 'VariableNamingRule','preserve');
PS = readtable(payloadSpectralFile, 'VariableNamingRule','preserve');
PV = readtable(payloadSolarFile, 'VariableNamingRule','preserve');

%% COLUMN GROUPS
UV_COLS  = {'UVA','UVB','UVC'};
VIS_COLS = {'V','B','G','Y','O','R'};
IR_COLS  = {'R2','S','T','U','V2','W'};

GROUND_VOLT_COLS  = {'Volt1','Volt2'}; % ground has 2 panels
PAYLOAD_VOLT_COLS = {'Volt1','Volt2','Volt3','VoltAvg'};  % payload has 3

%% PLOT 1: Payload altitude vs time
f = figure('Color','w','Name','Payload Altitude vs Time');
plot(pTime, altA, 'LineWidth', 1.6);
xlabel('Time');
ylabel('Altitude');
title('Payload Altitude vs Time');
grid on;
```

B.2 - Data Recovery

The following is an abridged version of the MATLAB code used for recovering the corrupted data.

```
%%OpenLog Data Recovery
clear; clc;

infile = 'corruptedData.txt';
outPrefix = 'recoveredData';

raw = fileread(infile);
u16 = uint16(raw);
```

```
%Split each 16-bit char into hi/lo bytes, swap order, remove nulls
hi = uint8(bitshift(u16, -8));
lo = uint8(bitand(u16, 255));
bytes = reshape([hi.'; lo.'], 1, []);
bytes_no_null = bytes(bytes ~= 0);
txt = native2unicode(bytes_no_null, 'latin1');
name  = 'recoveredData';
text  = txt;
```

Appendix C - Sample Data

C.1 - Ground System Sample

The following is sample data from the SD card for the ground system on launch day. The ground system data is representative of what is in both of the separate data files obtained for the payload.

Time,UVA,UVB,UVC,V,B,G,Y,O,R,R2,S,T,U,V2,W,Volt1,Volt2,VoltAvg,Temp1,Temp2
1.06,0.00000,0.00000,0.00000,10277.20703,34830.00000,8346.00000,8706.11035,11780.04687,10277.05175,10277.20703,34830.00000,8346.00000,8706.11035,11780.0468
7,10277.05175,5.82,5.82,5.82,24.00,24.00
,3.00,84.99871,13.23427,37.46183,8202.13574,34826.03125,8208.01953,8587.01562,11657.37597,10500.23535,8202.13574,34826.03125,8208.01953,8587.01562,11657.37
597,10500.23535,5.79,5.82,5.80,24.00,24.00
,4.93,84.99871,13.17803,37.12241,10788.78710,34830.03515,8234.25000,8603.03515,11880.01562,10552.00390,10788.78710,34830.03515,8234.25000,8603.03515,11880.
01562,10552.00390,5.79,5.82,5.80,25.00,25.00
,6.87,84.99871,13.00788,37.22626,12416.00000,35840.03125,8322.62500,8967.03515,12323.05468,10249.14160,12416.00000,35840.03125,8322.62500,8967.03515,12323.
05468,10249.14160,5.79,5.79,5.79,26.00,25.00

C.2 - Corrupted Data Sample

The following is a small snippet of data that was corrupted when OpenLog attempted to write to the SD card. This is an example of what was later translated back into readable data like in C.1.

T業攬啖秜啖箫啖繩噕箫䌷夬伬划刲ᓄ€€ᓆ㈬圬呥浰㉜ო哘浥ᘒ噁ᘒ噂ᘒ嘯ᓆᓄφᓇ€ᘓᑫᑫ㈬靣听唬嘲϶€鬶瀘ᘒ·嬙· ᔕᔕ °· ᔕᔕ °· ᔕᔕ °㊎ᔕ孽㐮
襃仩㊖㐮· ᔕᔕ °· ᔕᔕ °· ᔕᔕ °· ᔕᔕ °㊎ᔕ孽㐮襃仩㊖㐮· ᔕᔕ °· ᔕᔕ °· ᔕᔕ °· ᔕᔕ °ㄴ흐ㅇ」ロδパー· 卄ᅐ哭· ぐ嬙 °· ぐ㑉 °㊎ᔕ
孽㐮襃仩㊖㐮· ᔕᔕ °· ᔕᔕ °· ᔕᔕ °· ᔕᔕ °㊎ᔕ孽㐮襃仩㊖㐮· ᔕᔕ °· ᔕᔕ °· ᔕᔕ °· ᔕᔕ °ㄴ흐ㅇ」ロ┤哭· 卄㈰佳· ぐ嬙 °· ぐ㑉
°㊎ᔕ孽㐮襃仩㊖㐮· ᔕᔕ °· ᔕᔕ °· ᔕᔕ °· ᔕᔕ °㊎ᔕ孽㐮襃仩㊖㐮· ᔕᔕ °· ᔕᔕ °· ᔕᔕ °· ᔕᔕ °ㄷ°」ロ· 伆ᅷᅇ悐卄ᅷᅇ㊗仩ᅷᅇ㊔嬫