2019 LIFT PROJECT

PAYLOAD OPERATIONS MANUAL

COSMIC RADIATION PROJECT

Jeff Bell, Mary Block, and Garrett Burrows

WYOMING
NASA Space Grant Consortium

UNIVERSITY OF WYOMING

INTRODUCTION & PAYLOAD OVERVIEW

This manual serves as a record to the Cosmic Radiation Team's Project over the 2019 year. This project focused on measuring cosmic radiation (the Geiger counter could detect Beta particles and Gamma rays) as a function of altitude. An Arduino 101 was connected to a SparkFun Pocket Geiger and an Adafruit SD Datalogger; this unit, along with its power supply, was the primary source of data. High altitude balloons were used as the transport method to ferry payloads to the upper atmosphere (~30km above sea level).

Students from Newcastle High School were taught about radiation and asked to design and develop a shield to fit around their Geiger counter. Two teams were formed and over the course of a month the students worked towards their goal. After the successful construction of their radiation shields, both teams flew their payloads on a balloon launch.

OPERATING PROCEDURES

Payload operation is simple and only requires that the battery be plugged into the Arduino. This automatically turns on the Arduino and begins the process of data recording. The user can check that data is being recorded by looking for a flashing red LED on the SD card data logger. If the red LED is not flashing, the user can try gently shaking the Geiger counter which sometimes causes the Geiger counter to (erroneously) record data points (see *Known Issues & Troubleshooting*).

TECHNICAL DETAILS

1. Data

The data collected by this payload are from Beta particles and Gamma rays. The Geiger counter begins operating upon the battery being plugged into the Arduino. Every time the Geiger counter intercepts a Beta particle or Gamma ray, the SD card logs the time (in milliseconds) since the battery was powered, the intercept count, and the counts per minute. It also logs two Gieger counter measurements: the radiation dosage in terms of microsieverts per hour (μSv/h) and the radiation dosage error.

The radiation dosage is the measurement of most interest to this project. Tested background radiation dosages at the ground in Laramie (7,200 ft / 2220 m above sea level) were generally around 0.05–0.1 μSv/h. Higher in the atmosphere, radiation measurements from an unshielded Geiger counter can exceed 2.0 μSv/h, more than two orders of magnitude larger than at the ground.

With an effective shield, the Geiger counter would hypothetically intercept fewer cosmic radiation particles during balloon flight, thus lowering the overall radiation dosage. However, ineffective shields may actually *increase* the radiation dosage measured by the Geiger counter. This is likely the result of secondary radiation production that occurs when cosmic radiation particles interact with the shielding material itself, causing these particles to break down into even more particles, some of which decay into Gamma rays that more easily penetrate the shielding and reach the Geiger counter.

2. Hardware & Materials

(a) Description of Parts

The payload consists of an Arduino 101 system secured inside of a Styrofoam payload box. The individual parts of the Arduino system are listed below:

Part	Description / URL	Price*
Arduino 101	Buy online: SparkFun Part No. DEV-13787	Retired
Geiger Counter	Buy online: SparkFun Part No. SEN-14209	$69.95
Data Logger	Buy online: Adafruit Part No. 254	$7.50
MicroSD Card	Can buy at many stores or online, we used a SanDisk 16GB microSD card (no need to use cards higher than 16GB)	$5–10
Battery	We used two 7.4V 1150mAh LiPo battery packs connected in series to a 2.1mm center-positive barrel jack, giving us 14.8V	---
Wiring	Buy online: Adafruit Part No. 1957	$1.95
Payload Box	We purchase our Styrofoam boxes through StratoStar, with inner dimensions of 8” x 6” x 4” and outer dimensions of 11” x 9” x 7”	$30.00
3D-Printed Platform	We 3D-printed a small platform to securely attach the SD card logger to the Arduino	---

**Per unit, as of February 2020*

(b) Payload Configuration & Assembly

Once all components have been gathered, follow diagrams and instruction in the following websites to correctly assemble the setup:

- https://learn.adafruit.com/adafruit-micro-sd-breakout-board-card-tutorial/arduino-wiring
- https://github.com/gskielian/Arduino-DataLogging/tree/master/Bash-One-Liner

A small platform was designed in SolidWorks to be used as a shelf for the vital components during launch. The final assembly of the Arduino system is shown in Figure 1. In Figure 2, the Geiger counter has been shielded within one of the student-developed protective housings prior to launch. Finally, Figure 3 shows how the entire package was secured within the payload box.

Since we had two K-12 teams for this project, we flew both of their payloads together. To simplify comparison between the data collected by the payloads, we connected the two payload boxes horizontally so that both shielded Geiger counters would always be at the same exact altitude during flight (see Figure 4). This also removed any possibility of interference if one of the boxes was situated above the other.

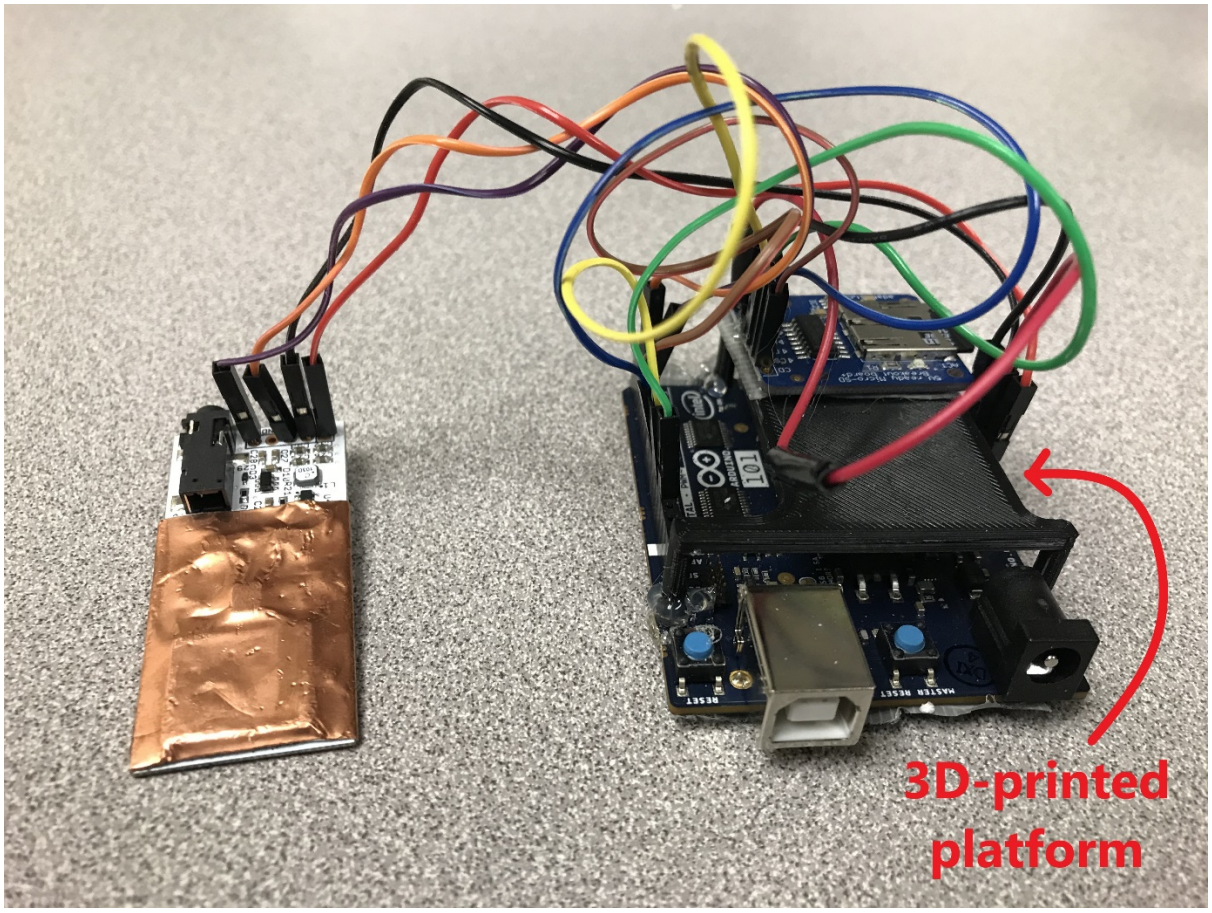

Figure 1 - Arduino and data logger (right) with attached Geiger counter (left)

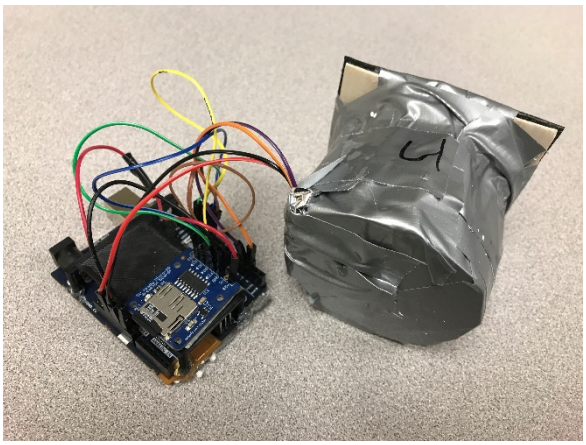

Figure 2 - Arduino assembly with Geiger counter wrapped in protective shielding

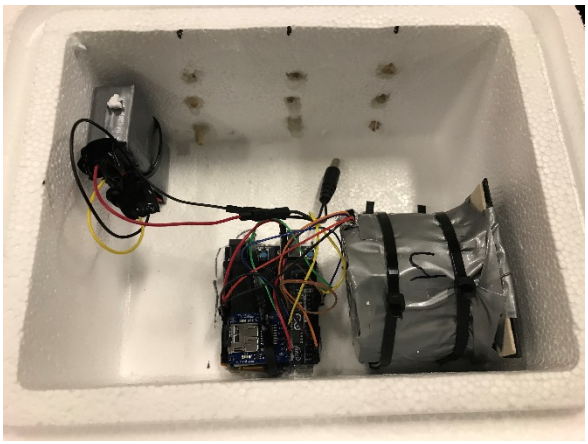

Figure 3 - Arduino assembly (including battery) secured within a payload box prior to launch

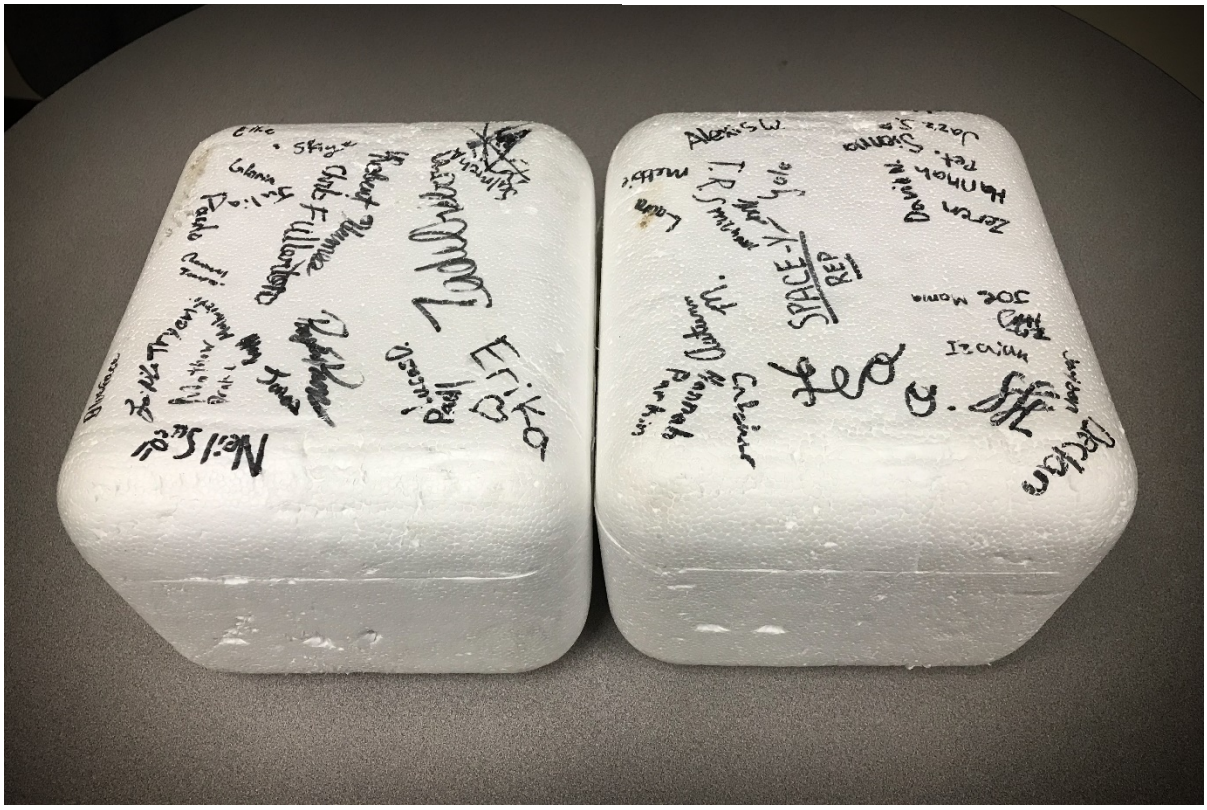

Figure 4 - Dual-box setup used for the launch

3. Software & Coding

The Arduino IDE (https://www.arduino.cc/en/main/software) was used as the main platform for developing code. An open source library titled “Radiation Watch” was discovered and downloaded. It contained a variety of useful Arduino files including one titled “SdCardCsvLogger.” This code was only modified by changing the SdCard Pin from 4 to 10. Pin 10 was the right pin output for the Arduino 101 but the correct pin will vary from model to model. See Appendix A for the exact code.

4. Additional Materials

Other materials were indirectly used for the payload as well, either during payload design and construction or during the development of the payload shielding. They include the following:

Part	Description / URL	Price*
Solder	Also need access to a soldering iron	---
Hot Glue	Can buy at any craft store	---
Zip Ties	Used to secure Arduino system to payload box	---
Steel Sheet	Buy online: Amazon Link	$10.48
Galvanized Steel Sheet	Buy online: Amazon Link	$10.98
Aluminum Sheet	Buy online: Amazon Link	$9.88
Balsa Wood	Buy online or at a local craft store	---
Space Blanket	Buy online: Amazon Link	$3.95
Radiation Source	To test, it is helpful to have a radiation source that emits Beta particles, such as a radium sample (see safety considerations)	---

**Per unit, as of February 2020*

CLASSROOM IMPLEMENTATION

We posed our experiment in the form of a space race between two competing nations. This was done to produce some friendly competition, as well as get the students excited about the experiment. We assigned the students roles based on survey responses on who would be best fitted for each role. The roles include Space Agency Director, Engineer, Computer Scientist, Economist, Public Relations, and Research and Development.

We had two work days, wherein we introduced the experiment, including slides and presentations, and asked the students to begin brainstorming their payload ideas. The LIFT fellows were present to guide the students and answer any questions the students may have had. The second work day was comprised of building the payloads. Since neither of our teams finished their payload design within two meetings, they were allowed to finish their payloads in the classroom without the LIFT fellows being present.

The third day was launch day. There was no lesson included on the launch day. The third and final lesson was conducted over Zoom, since it was only reviewing the data and conclusions. This took only about 20 minutes, so there was no need to travel to Newcastle for the last lesson.

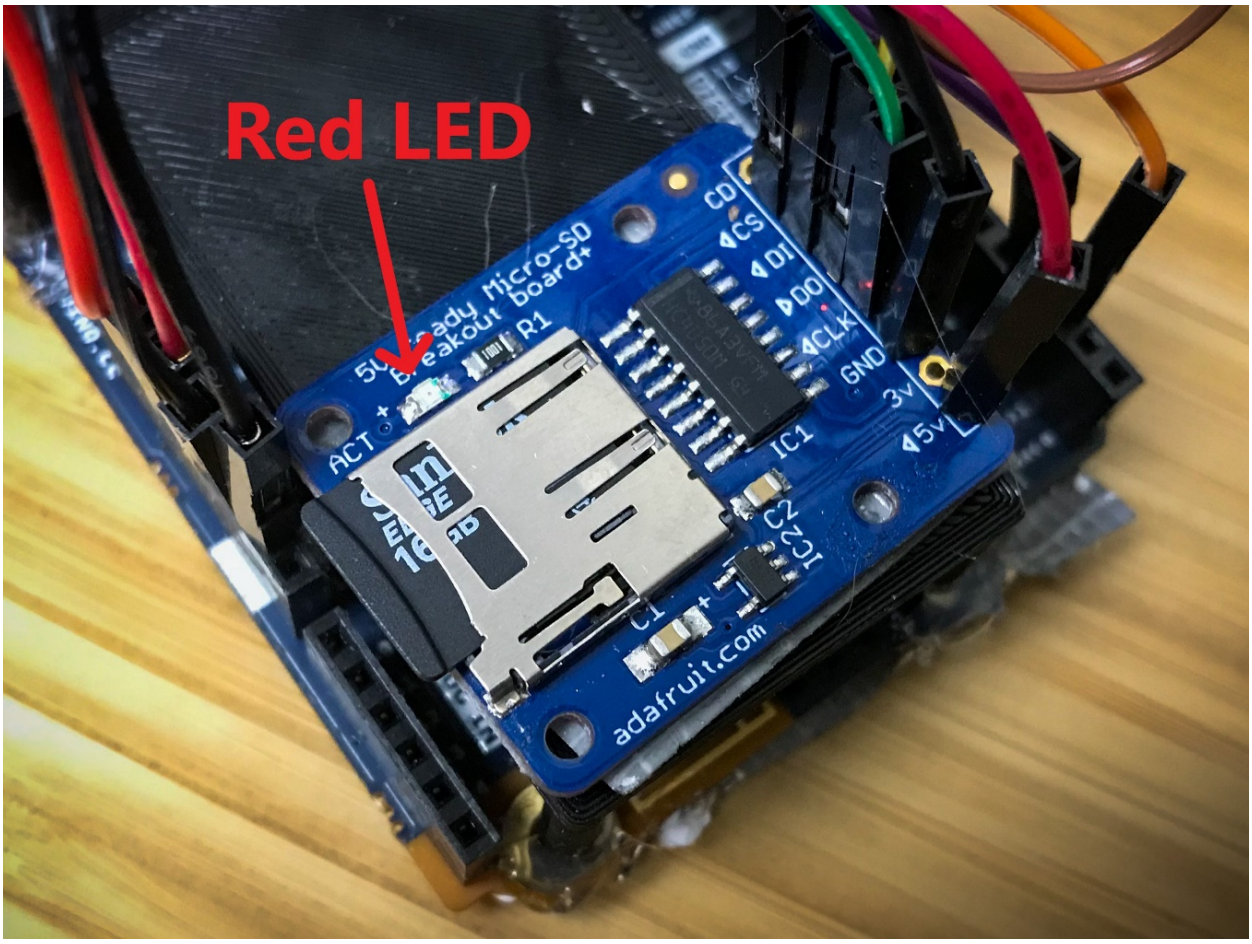

Figure 5 - Closeup of SD card data logger and the location of the red LED light

KNOWN ISSUES & TROUBLESHOOTING

- When naming the output file in the code, keep the string length as short as possible (e.g., “data.csv”). In the past, long string names resulted in unreadable data.
- If the red LED light on the SD card data logger (see Figure 5) is not flashing after the battery is plugged in, try gently shaking the Geiger counter. This can cause the Geiger counter to register a false “detection.” If so, the LED will flash, revealing that the Geiger counter is indeed functioning. These false detections are logged on the SD card but do not actually contribute to the radiation dosage measurements.

SAFETY CONSIDERATIONS

- Use precautions when working with the radiation source. Our sources were two radium watch-hand samples. If your source has more radioactive potential than the one we used, please resort to the guide in the following link:

https://admin.kuleuven.be/sab/vgm/kuleuven/en/riskactivities/rp/RadiationProtectionPrecautions.html

- Use precautions when soldering. Don't inhale fumes, be careful not to burn yourself or your partner(s), make sure the cleaning sponge is properly moistened, and be sure to properly clean the soldering iron after enough solder has accumulated. An alternative solution is to use a brass pad specifically designed for cleaning the soldering tip.
- When working with metal, tape the sharp sides and/or work with gloves. This is for your safety and student safety. This is so you don't cut yourself on the metal samples.
- When working with students, be sure you cover basic safety precautions.
- IMPORTANT: We strongly recommend that you DO NOT bring the radiation source to the classroom!

NOTES

- It is recommended that a more powerful microprocessor be used for this experiment, such as an Arduino Mega. The Arduino 101 was sufficient to run the Geiger counter and SD card logger, however additional components caused a malfunction due to a shortage in processing power.
- It is recommended that the SD data logger be connected to the largest voltage output on your Arduino as long as the output is not larger than 5V. On the Arduino 101, reserve the 5V output for the data logger and use the 3.3V output for the Geiger counter. If you want to use the 3.3V output for the data logger, make sure that it does not draw too much current.
- Another recommendation is to use an ethernet shield or microSD shield with your Arduino instead of a standalone SD data logger. This helps organize the setup and keeps the 5V and 3.3V outputs open for the Geiger counter and another component such as a real-time clock for adding date and time stamps to the data.
- Ensure no electrical shorts are present in the wiring before powering the Arduino.
- Ideally, you would also fly a control payload as well. This would require a total of three Arduino setups if the students are also competing in the space race scenario.

ACKNOWLEDGEMENTS

This work was funded through National Science Foundation Grant DUE-1821566.

APPENDIX A: *Radiation Watch* Arduino Code

```
 // You can change here the history length to reduce the memory footprint.
#define HISTORY_LENGTH 200
#include "RadiationWatch.h"
#include <SD.h>
#include <SPI.h>
/* Same as SerialCsvLogger, but store the CSV values to an SD card. */

RadiationWatch radiationWatch;
const byte sdCardPin = 10;
File dataFile;
unsigned long csvStartTime;

void csvKeys(Print &print=Serial)
{
  // inizialise start time
  csvStartTime = millis();
  // Write CSV keys to a print class (default print class is Serial).
  print.println("time(ms),count,cpm,uSv/h,uSv/hError");
}

void csvStatus(RadiationWatch radiationWatch, Print &print=Serial)
{
  // Write CSV to a print class (default print class is Serial).
  // time(ms)
  print.print(millis() - csvStartTime);
  print.print(',');
  // count
  print.print(radiationWatch.radiationCount());
  print.print(',');
  // cpm
  print.print(radiationWatch.cpm(), 3);
  print.print(',');
  // uSv/h
  print.print(radiationWatch.uSvh(), 3);
  print.print(',');
  // uSv/hError
```

```
  print.print(radiationWatch.uSvhError(), 3);
  print.println("");
}

void onRadiationPulse() {
  // For each radiation write the current values to the SD card in CSV.
  csvStatus(radiationWatch, dataFile);
  dataFile.flush();
}

void setup()
{
  Serial.begin(9600);
  // SD setup.
  if(!SD.begin(sdCardPin)) {
    Serial.println("SD card init failed, or not present");
    return;
  }
  dataFile = SD.open("rad.csv", FILE_WRITE);
  if(!dataFile) {
    Serial.println("Failed to open file");
    return;
  }
  radiationWatch.setup();
  // Print the CSV keys and the initial values.
  csvKeys(dataFile);
  csvStatus(radiationWatch, dataFile);
  dataFile.flush();
  // Register the callback.
  radiationWatch.registerRadiationCallback(&onRadiationPulse);
}

void loop()
{
  radiationWatch.loop();
}
```