Recreating Bouncing Signals for Cosmic Ray Detection

By Eric Hu, Professor Stalerman, Professor Armendariz

Who am I?

Eric Hu

High School: Brooklyn Technical High School

Interests: Robotics, Programming, Engineering

Brief Project Overview

Cosmic Ray Detection with Arduino

What are Cosmic Rays?

High energy particles that originate mostly from other stars.

Why are we looking for cosmic rays?

Cosmic Rays come at relatively constant intervals. Therefore, if we suddenly stop detecting these rays or if there are dips in cosmic ray detections, we know that something strange is happening. We can then direct our attention to finding what caused the fluctuation in cosmic ray data to learn new information.

Brief Project Overview

Satellite

Sends time data that allows us to situate our muon data in time

GPS Sensor

Communicate with the satellite to get positioning, time, temperature and other atmospheric data

Glass Sheet (Scintillating Material)

Detect muon data which is used to find cosmic rays

Photosensor

Looks at the Glass Sheet for any light emissions

Arduino

Read the data from the GPS sensor, then display a graph and interpret the data to detect cosmic rays

Brief Project Overview

Current Problem

Bouncing problem: sometimes, signals from the GPS sensor are read more times than they are supposed to. Single signals are read multiple times.

Task

Determine if the problem exists for simpler programs and recreate the bouncing problem and try to fix it if it does exist.

Method

Test the basic Interrupt Arduino function with many types of inputs to determine if the problem is rooted in Arduino hardware

Tested Inputs

- Physical Button to create a square wave
- Attaching a PPS (once per second) signal from the GPS module and use that square wave
- Creating a square wave from the Arduino and reading it with its own Interrupt pin

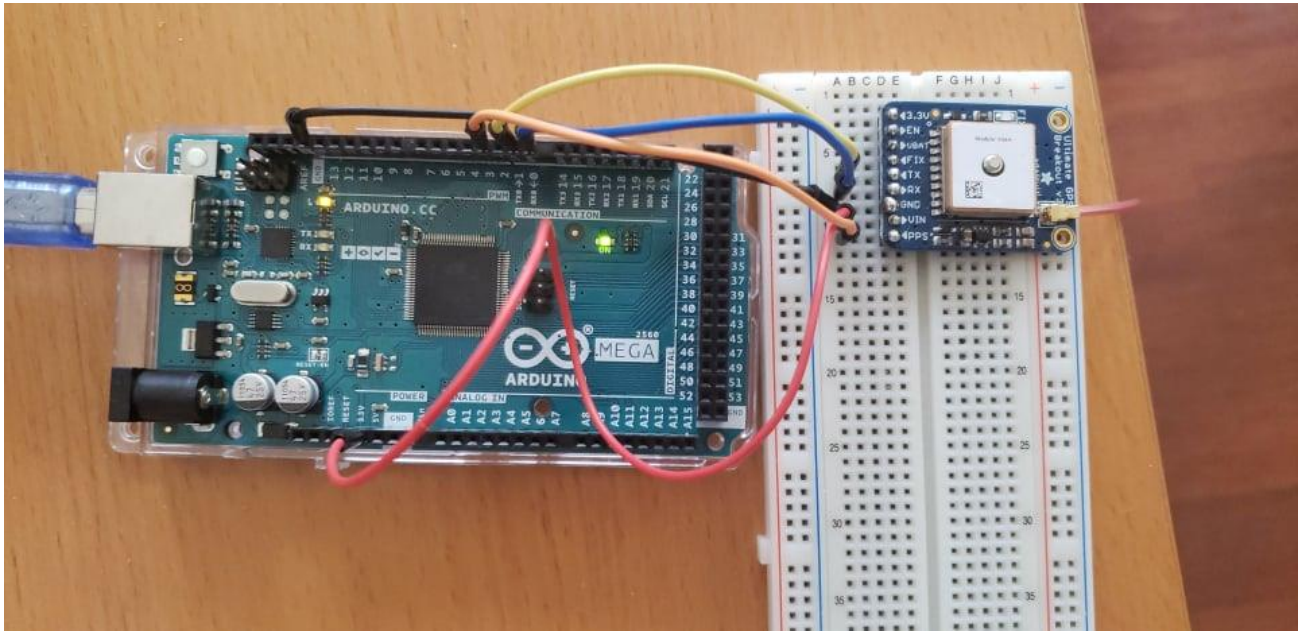
Method

Test the basic Interrupt Arduino function with many types of inputs to determine if the problem is rooted in Arduino hardware

Measurement methods

- Count number of times the Arduino registers HIGHs and LOWs
- Record the timestamps of when the Arduino registers HIGHs and LOWs to find fluctuations between the periods of each pulse
- Look at the graph

Arduino Setup



Arduino Code

Physical Button to create a square wave

```
void setup() {
  pinMode(3, INPUT);

  Serial.begin(9600);

  attachInterrupt(digitalPinToInterrupt(3), printOne, CHANGE);
}

int buttonState = LOW;

void loop() {
  Serial.println(buttonState);
}

void printOne() {
  buttonState = !buttonState;
}
```

Arduino Code

Attaching a PPS (once per second) signal from the GPS module and use that square wave

```
void setup() {

  Serial.begin(9600);
  pinMode(3, INPUT);
  attachInterrupt(digitalPinToInterrupt(3), received, CHANGE);
}

void loop() {
  //Serial.println(digitalRead(3));
}

void received() {
  Serial.println(digitalRead(3));
}
```

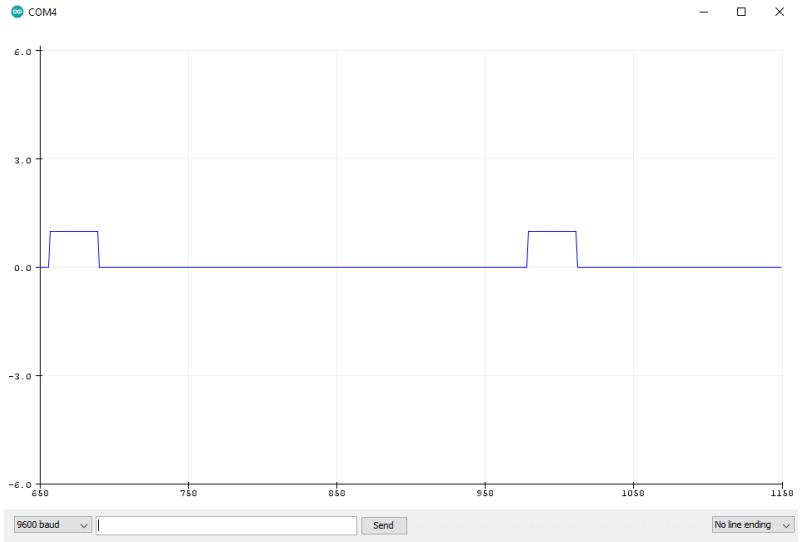
Arduino Code

Creating a square wave from the Arduino and reading it with its own Interrupt pin

```
void setup() {
  pinMode(13, OUTPUT);
  pinMode(3, INPUT);

  Serial.begin(9600);

  attachInterrupt(digitalPinToInterrupt(3), received, CHANGE);
}

int pulseState = HIGH;
int receiverState = LOW;

void loop() {
  maybeChange(100000, 1000000);
  Serial.println(pulseState);
//  Serial.print(",");
//  Serial.println(receiverState);
}

long t; //time
long pt = 0; //previous time
long timeLow;
```

```
void maybeChange(long pulseWidth, long
period) {
  t = micros();
  timeLow = period-pulseWidth;
  if (pulseState == HIGH) {
                if (t-pt > pulseWidth) {
                pulseState = !pulseState;
                pt = micros();
                }
  } else if (pulseState == LOW) {
                if (t-pt > timeLow) {
                pulseState = !pulseState;
                pt = micros();
                }
  }
}

void received(){
  receiverState = !receiverState;
}
```

Data Collected from Arduino



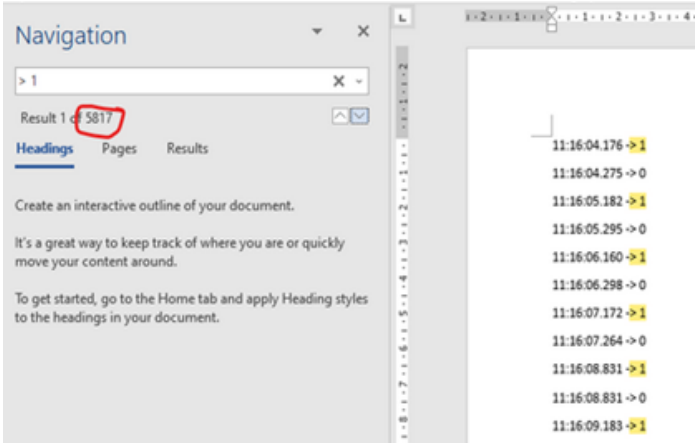
Raw PPS data from the GPS

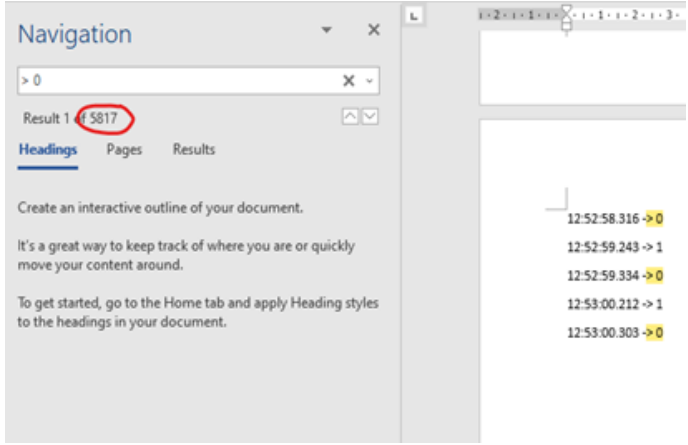
11:16:04.176 -> 1
11:16:04.275 -> 0
11:16:05.182 -> 1
11:16:05.295 -> 0
11:16:06.160 -> 1
11:16:06.298 -> 0
11:16:07.172 -> 1
11:16:07.264 -> 0
11:16:08.831 -> 1
11:16:08.831 -> 0
11:16:09.183 -> 1
11:16:09.275 -> 0
11:16:10.184 -> 1
11:16:10.277 -> 0
11:16:11.196 -> 1
11:16:11.287 -> 0
11:16:12.200 -> 1
11:16:12.255 -> 0
11:16:13.176 -> 1
11:16:13.269 -> 0
11:16:14.194 -> 1
11:16:14.284 -> 0
11:16:15.156 -> 1
11:16:15.295 -> 0
11:16:16.173 -> 1
11:16:16.266 -> 0
11:16:17.185 -> 1
11:16:17.278 -> 0
11:16:18.178 -> 1

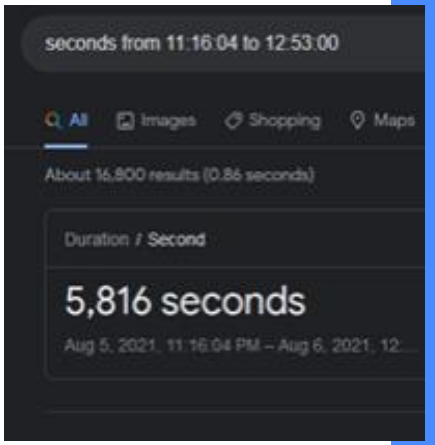
Analyzed code for when the signal changes from high to low

Findings

I let my Arduino run for around 1h30 and saved the data of when the signal changes from low to high. Then, I counted how many 1s and 0s there were and compared that number against how many seconds elapsed during the time period to determine if the signal bounced. Since the numbers are the same, the signal didn't bounce.





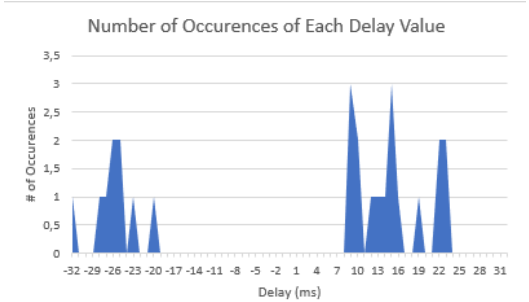


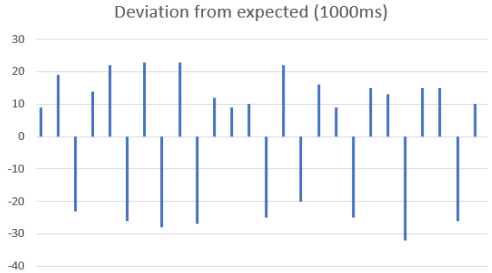
(There is 1 less second than 1s and 0s because Google is showing the seconds between the timestamps but the Arduino printed out a 1 and 0 on the last second)

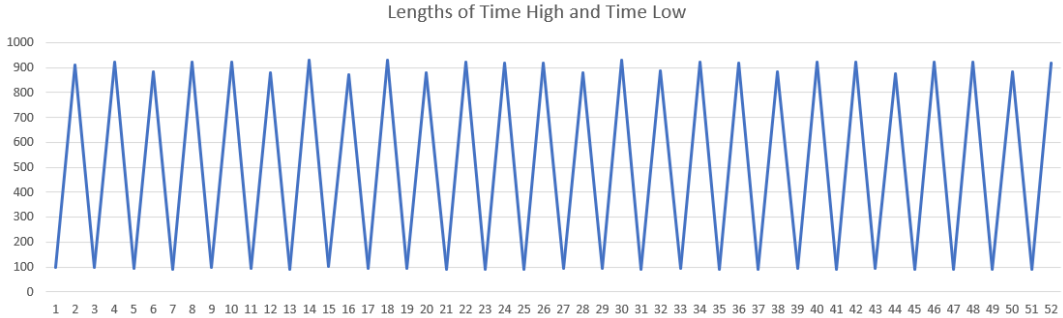
Findings

Comparing the timestamps of when the data changed, we found that the Arduino is relatively inaccurate, being off by milliseconds for each pulse. However, it doesn't look like that was caused by the signals being read multiple times.

	A	B	C	D	E
1	Value	Timestamps	ms Diff (with next line)	Deviation from expected (1000ms)	Difference between consecutive pulse periods
2	1	13:10:44.058	98	9	
3	0	13:10:44.156	911		10
4	1	13:10:45.067	97	19	
5	0	13:10:45.164	922		-42
6	1	13:10:46.086	94	-23	
7	0	13:10:46.180	883		37
8	1	13:10:47.063	92	14	
9	0	13:10:47.155	922		8
10	1	13:10:48.077	100	22	
11	0	13:10:48.177	922		-48
12	1	13:10:49.099	93	-26	
13	0	13:10:49.192	881		49
14	1	13:10:50.073	92	23	
15	0	13:10:50.165	931		-51
16	1	13:10:51.096	102	-28	
17	0	13:10:51.198	870		51
18	1	13:10:52.068	93	23	
19	0	13:10:52.161	930		-50
20	1	13:10:53.091	94	-27	
21	0	13:10:53.185	879		39
22	1	13:10:54.064	91	12	
23	0	13:10:54.155	921		-3
24	1	13:10:55.076	90	9	
25	0	13:10:55.166	919		1
26	1	13:10:56.085	91	10	
27	0	13:10:56.176	919		-35
28	1	13:10:57.095	94	-25	
29	0	13:10:57.189	881		47
30	1	13:10:58.070	93	22	
31	0	13:10:58.163	929		-42
32	1	13:10:59.092	92	-20	
33	0	13:10:59.184	888		36



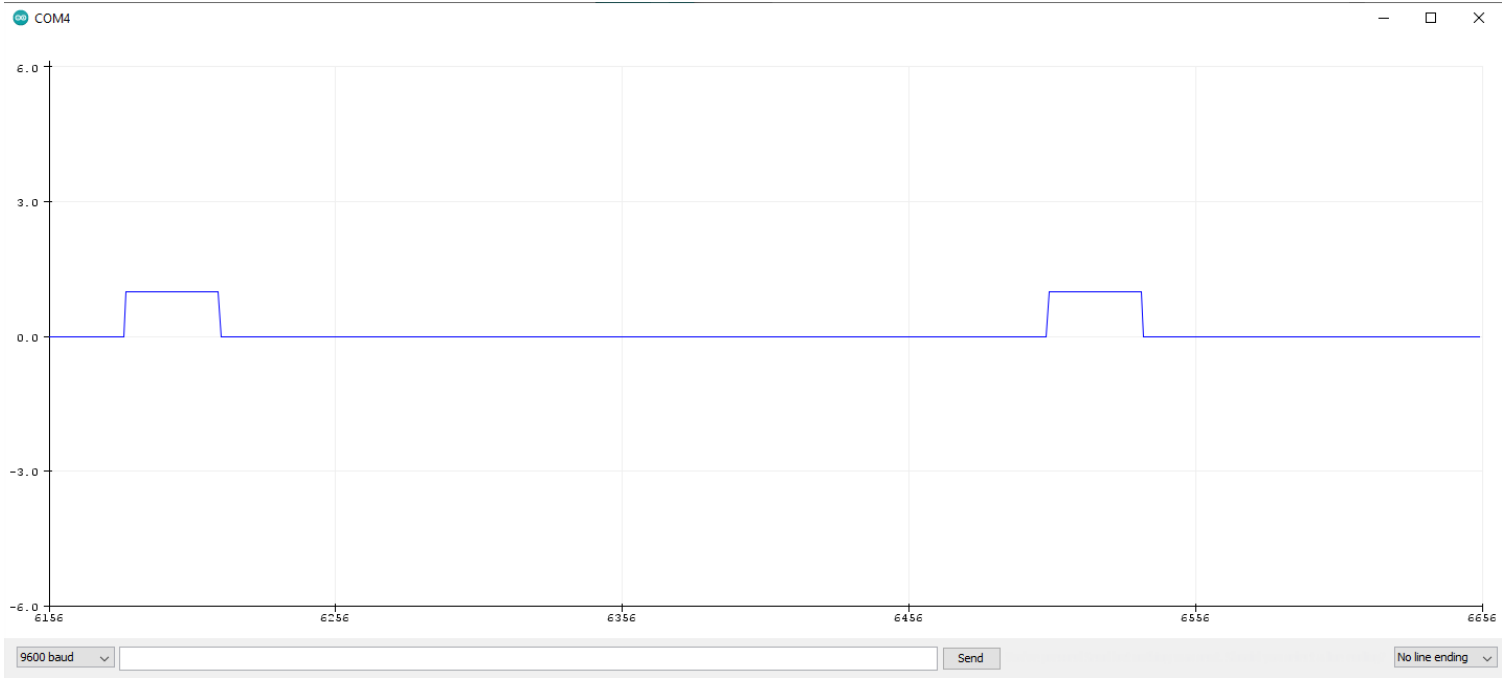




There looks to be no signal bouncing on this graph

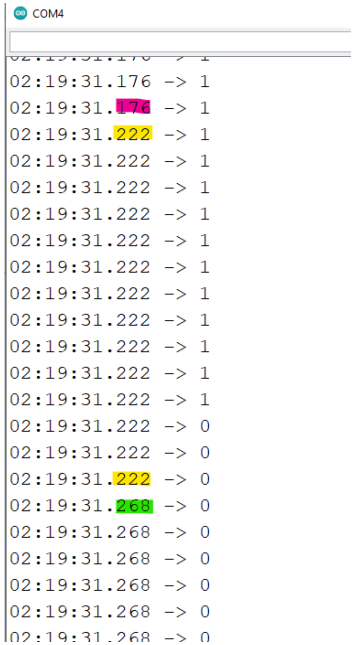
Findings

Finally, visually looking at the graph, there also seems to be no bouncing there.



Discussion

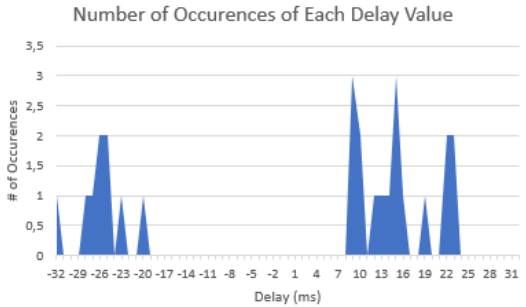
Interesting Observations

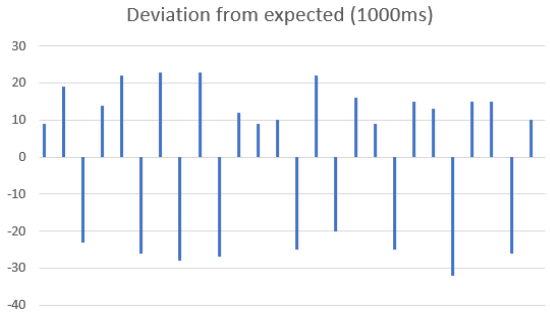


The Arduino timestamps seem to be updating at certain intervals and doesn't show the time in between. This might have affected the data from when we tracked the timestamps, but we are not sure if this only happens when data is continuously being printed out, which would not be the case in our experiment.

Discussion

Interesting Observations





The period length deviation from the expected 1000ms seemed to be either ≥9ms or ≤-20ms, and we don't know what causes that.

Conclusion

How the results connect with the problem

Because we didn't find any bouncing from the Interrupt function itself or any input going into the Arduino, we know that the bouncing problem occurred in other parts of the Arduino code originally used to track cosmic rays.

Conclusion

Future Experiments

- Continue the testing with more of the original code in order to determine what caused the signal bouncing.
- Determine where the output inaccuracy measured from the Arduino is coming from, and whether it affects our data collection.
- Find other methods to record cosmic ray data without using Arduino interrupts.

Acknowledgements

Special thanks to:

- **Professor Stalerman and Professor Armendariz for guiding me through this project**
- **Professor Marchese for giving me the opportunity to conduct research**
- **Lamisa, Fatimah, William and Fabio for doing this research alongside me**