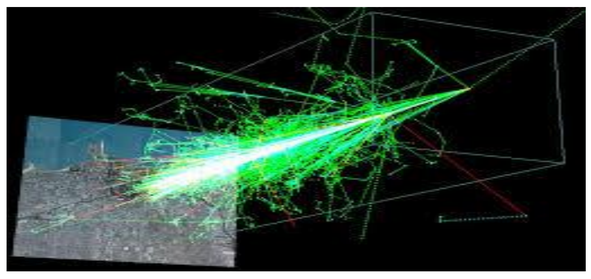
Enabling data collection for cosmic ray detector

Junjie Chen(city college of New York, New York, NY 10031)
Mentor: David Jaffe (Brookhaven National Laboratory, Upton, NY 11973)
Collaborators: Raul Armendariz(Queensborough Community College, Bayside, NY 11364)
Aiwu Zhang(Brookhaven National Laboratory, Upton, NY 11973)

Abstract:

A cosmic ray shower occurs when a cosmic ray interacts in the upper atmosphere. Many of the muons from the resulting shower penetrate the atmosphere and strike the earth's surface. Our detector is designed to create pulse when muons pass through the detector. Our Arduino board will receive a signal trigger when there is a muon coming in. All the data including the peak voltage, GPS(Global Positioning System), temperature and pressure is collected for the trigger. However, the GPS time is not accurate enough since it takes time to decode the signal (NMEA sentence) from GPS. So, PPS(Pulse Per Second) which indicates the start of an integer second is used to correct the error with an inner clock of Arduino. We also write two programs to automatically receive and save the data as texts file on the Raspberry pi 3 and automatically upload those files into our storage space.



This is a Cosmic ray shower
(https://en.wikipedia.org/wiki/Air_shower_(physics))

Time stamping :

1, Instruments :

Arduino mega2560, Adafruit GPS receiver(PA6H1F1752).

2, Method:

In the time stamping process, we trust the PPS signal from GPS receiver, which means the PPS signal is the start of new second.

The decimal place of the time is calculated by a 16MHz internal clock of Arduino (Timer 1), which means we can only make the accuracy at most 62ns.

$$t = second + \frac{counts}{f}$$

The Timer 1 counter in Arduino mega2560 is a 16bit counter that counts the Arduino clock, which means it will overflow every 4ms. To use it to measure the decimal place of the time, we have to record the times of overflow. So, an ISR interrupt should be called:

$$\boldsymbol{TIMSK1 = \_BV(TOIE1);//enable\ the\ ISR}$$

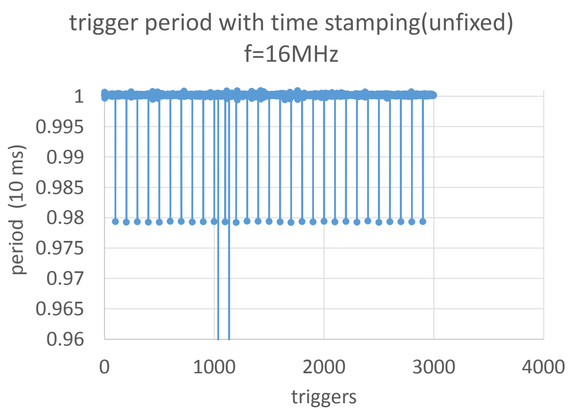
and

$$ISR(TIMER1\_OVF\_vect)\{ \\ overflows++;\}$$

(Those code is direct use of the physical address of TIMER1)

3, Discussion

However the Arduino frequency is just around 16 MHz . If we use the inaccurate frequency, this is the result when we measuring the leading edge of 100 Hz pulse:



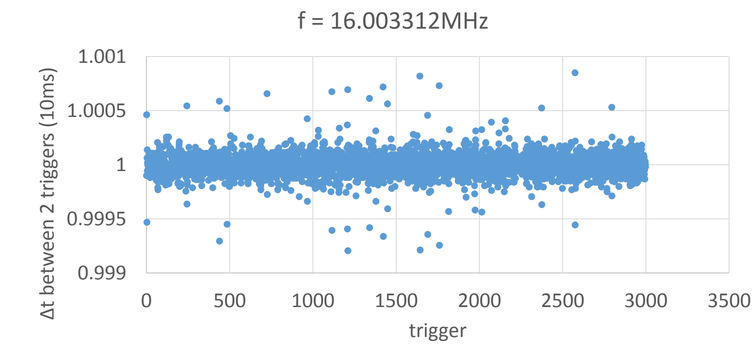
You can see each second, there is a drop on the period, which means the time stamping is fixed by the PPS signal. So, we have to measure the counts between two PPS signal which is the frequency.

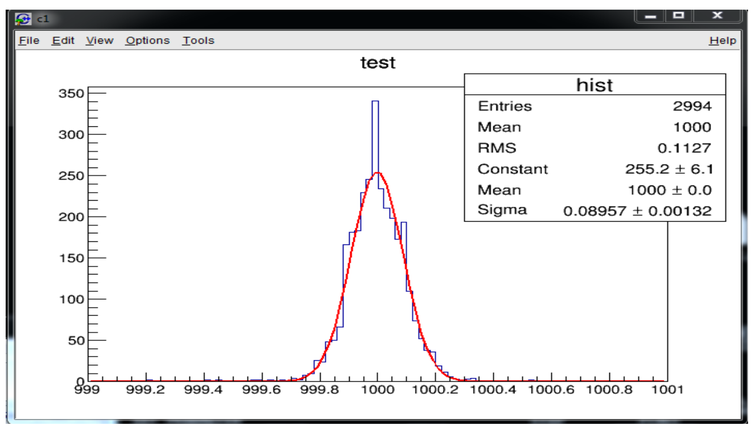
This is caused by :(count is the Arduino count since last PPS)

$$\Delta t = \left(sec2 + \frac{count2}{f(not\ fixed)}\right) - \left(sec1 + \frac{count1}{f(not\ fixed)}\right) \\ = 1 + \frac{count2 + f(real) - count1}{f(not\ fixed)} - \frac{f(real)}{f(not\ fixed)}$$

$$\Delta(\Delta t)(the\ drop\ at\ the\ end\ of\ the\ second) = 1 - \frac{f(real)}{f(not\ fixed)}$$

Then the period of 100Hz pulse is stable like the next two plot:





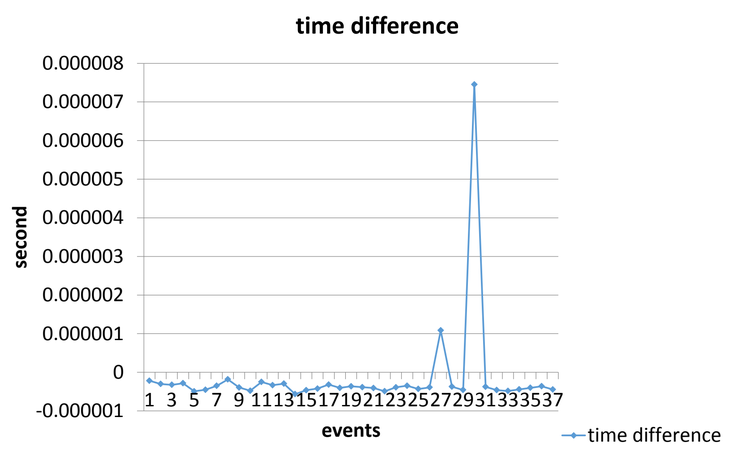
In order to record the signal as soon as the signal coming in, the PPS and events signal should also call interrupts. When PPS comes in, record the counts(for calculate the frequency) and reset it to 0. Since it takes time, we should measure that and add to the counts for events.

```
void ppS(){
  temp = TCNT1;
overflows2=overflows;
  TCNT1=0;
  overflows = 0;}
```

(TCBT1 is the physical address of our Timer 1 counter)
Then when the events comes, just record the counts.

```
void Trigger(){
  temp = TCNT1;
o=overflows;}
```

TCNT1 is the physical address for the Timer 1 counts.
To measure the accuracy of the time stamping, we do the next thing:
We use two Arduino boards to measure the same pulse from the generator, this is the results(cable length are almost same):



It's not good as what our expected like 62ns.

Here is the 3 possible reasons:

1, One of the PPS signal is not accurate because we see that one GPS receives 9 satellite signals but the other one only receives 6.
Since GPS receiver is solving inequation while correcting the time. So, more satellites , the more accurate time we will have. In order to reach the level of ns, we suggest to ensure all the GPS receiver receives around 8 satellites.

2, The Arduino stability for counts.

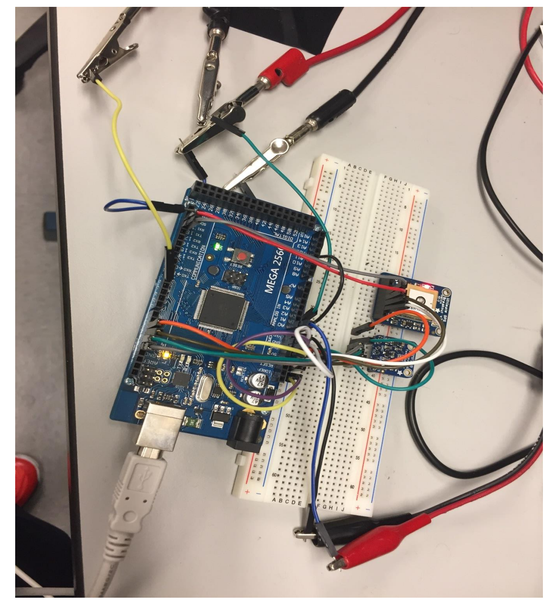
3, Time consumed on Arduino code.

4,Conclusion:

We understand time stamping of GPS for Arduino and we have the program to do the time stamping.

Most of the time difference on two Arduino board for the same pulse are within 500ns except two points.

So, the minimum gate for date analysis will be 500ns. (If the time difference are within 500ns, then we thought they are from a same shower)



Auto-uploading stage :

Tool: processing 3.0

We want to upload on Raspberry Pi for financial reason (we plan to build 30 detectors). So, we want to use raspberry pi to upload data to Dropbox.

However, there is no raspbian version Dropbox product. So, we write a program (cron_drop) to upload.

Meanwhile, since on the raspbian, we cant upload a file when that file is written in data. So, auto-copy should be done on the commend line.

Then, on one computer, we will use the commend line to auto-rename the data file due to the time and date.

This is part of the data collection code to write the data in text file on processing 3.0:

```
void serialEvent (Serial myPort) {
  // get the ASCII string:
  String inString = myPort.readStringUntil('\n');
  if (inString != null) {
  inString = trim(inString);
    output.println(inString);
    output.flush();
  }
 }
```

Date analysis program:

Tool: C language
After we collect all the data, we write a program to find the cosmic ray shower from those date.

There is a gate which is calculated by

$$gate = \frac{distance\ between\ detectors}{c}$$

So. If the distance is 30 km, then the gate will be 100us.

If most of the detectors receive a signal within the gate, then, it's a shower.

We will compare the date line by line so, we should use this function:

void getaline(FILE*r,int n) we defined in our code where *r is the pointer of the file and n is marking which file we are reading.

The complexity of the code will be N, since it only goes over each date file once.

Future work:

We need to figure out the reason for the error on time stamping(500ns). Measure the time consumed on each step of the Arduino program.

Reconstruction of the shower:
After we collect the location, time and energy information.

two methods:
1, Use the time and location of arrival on detectors and the uncertainty of the time stamping to make several inequations to find a possible range of original oscillation point.

2, Use the energy distribution to find the mid point and the angle of the cosmic shower.

U.S. DEPARTMENT OF ENERGY | Office of Science
BROOKHAVEN NATIONAL LABORATORY
NEW YORK STATE COLLEGIATE SCIENCE & TECHNOLOGY ENTRY PROGRAM CSTEP preparing science, technology & licensed professionals since 1986
NSF