Enabling data collection for cosmic ray shower detector

Junjie Chen

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. Some noise will also be caught. A discriminator and an and-gate are used to select the pulse. The discriminator will select the pulse whose voltage is higher than the setting. We put one detector on the top of the other one. So, if a muon passes through, both of the detectors will send a pulse to the and-gate and stimulate a square trigger pulse. Our Arduino board will receive the trigger as a signal that there is a muon coming in. So, 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 Timer 1 of Arduino. We also write two programs to automatically receive and save the data as text file on the Raspberry pi 3 and automatically upload those files into our storage space. By that, we are planning to have more than 20 detectors in different colleges and we are able to read all the data in our lab. From this project, I have gained valuable about the Arduino and processing programing, the GPS UTC time correction method and port communication.

Introduction:

The aim of this program is to build hundreds of cosmic ray shower (muon) detector. Those detectors will be placed at different places and keep recording data for find cosmic ray shower. My job in our group is write code to collecting data from our detector and do time stamping on those data with the UTC time on Arduino. Those data will saved as text file in Raspberry Pi. I also built an auto-uploading stage which is necessary since we plan to build hundreds of them which means we will have hundreds of Raspberry Pi. Finally, I wrote the code to do the data file analysis to find the coincidence shower.

Progress:

(1). Time stamping:

The time stamping is the most important part in the code on data collection since the muon is travelling in light speed and few milliseconds delay will cause huge difference in position. Our accuracy for the time stamping is 500ns. Many things we do to achieve that accuracy.

a. **PPS (Pulse per second)**
   is a high voltage signal coming from the GPS receiver. It indicates the start of a new second. The reason we use the PPS signal but not directly read the time from GPS is that: the GPS receiver usually send NMEA sentence and decoding for the NMEA sentence takes long time like 0.2 s.

   What we do is keeping reading the NMEA sentence (5 Hz) and parsing it, and setting the PPS signal as an interrupt. So, when the PPS signal comes, the interrupt function will be called. In this function, we will read the newest parsed second (NMEA second). And NMEA second +1 will be the actual time of this moment.

   The reason is the frequency of NMEA sentence is 5 Hz and the newest NMEA sentence is probably in the form like NMEA senonds.700. Since it takes around 0.2 second to decode that, you actually receive and parse the NMEA sentence at time around NMEA senonds.900. And then at the integer second NMEA seconds +1, the PPS comes.

   Now, we have the integer part of our events. Every event comes after that PPS signal will use that number as integer part of the second. We use an internal clock of Arduino mega 2560 to count the decimal place of the time.

b. **Arduino internal clock (Timer 1)**

   The name of the internal 16 bits clock is TIMER 1. It follows the Arduino frequency to count. So, its frequency is 16 MHz, which means it counts once around every 62.5ns.

The counts between the PPS signal and the events should be recorded. However, since the clock only has 16bit storage space, the counts will return to 0 around every 4ms ($62.5ns * 2^{16} = 4.096ms$). We call this situation overflow. The number of overflows between PPS and events should also be recorded.

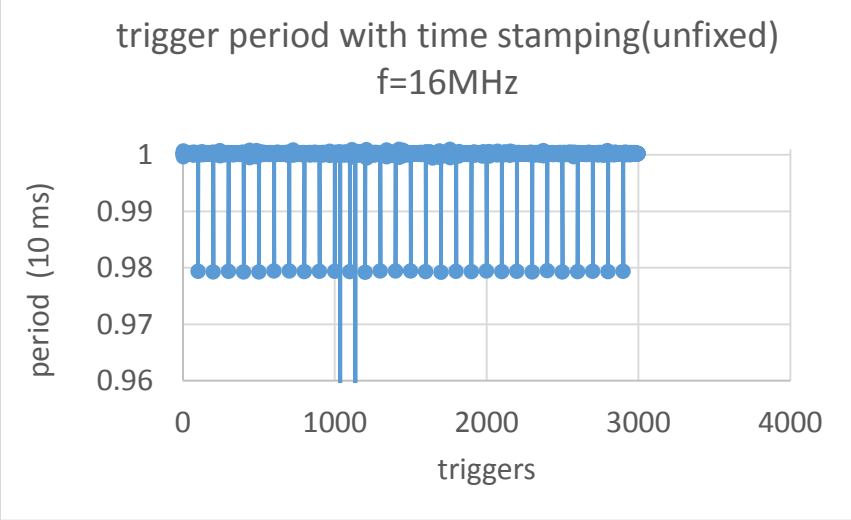
So, the overall situation will be like within PPS interrupt function, the TIMER 1 counts and overflow number will be zeroed. And record the counts and the overflows when the events come. Finally, use counts and overflows to calculate the real counts between PPS and events.

After we get the real counts, it needs to be converted into time by

$$time = GPS.seconds + 1 + \frac{counts}{f}$$

Where f is the frequency of the Arduino board. However, the frequency is just around 16MHz and it will have thousands counts difference.

This is what the time stamping will looks like when the wrong frequency is used:



Each of the drop point represents the time stamping is fixed by the PPS signal.

This is a calculation about how larger the drop will be:

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

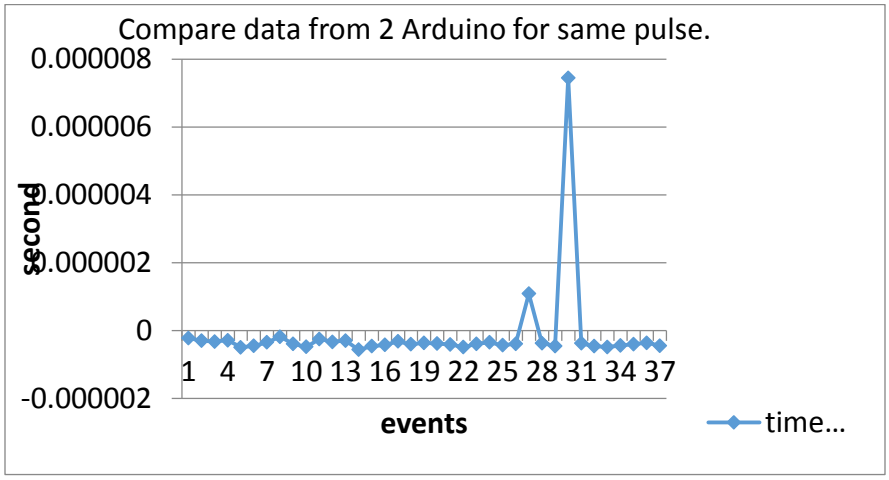
$$= 1 + \frac{count2 + f(real) - count1}{f(not\ fixed)} - \frac{f(real)}{f(not\ fixed)}$$

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

So, we have to measure the actual frequency. Since we already have the PPS signal, we just record the counts between two PPS signal which will become the frequency.

c. Time stamping ability test

We test our Arduino board ability to do time stamping by using two Arduino board with two GPS to measure the same pulse from pulse generator. The results is like this:



Most of the time difference is within 500ns except 2 points. They have like 5us difference.

My hypothesis for the reasons that cause this kind of difference is:

1, The 500ns difference, since one of the Arduino board is always ahead to the other one, might be caused by PPS signal coming at difference time.

2, For the 5us difference, it might be caused by the Arduino board or the Arduino physical property.

This need to be figured out in the future.

(2). Auto-uploading to the storage space (Dropbox)

After the data files are saved in the Raspberry Pi, we need to upload those into our Dropbox. Unfortunately, Dropbox does not have a product for Raspbian (Operation System for Raspberry Pi), so it is hard to do auto-uploading.

We find a software called cron_drop that can upload data file every midnight. However, if the date file is being written in by Arduino, it can't be uploaded on Raspbian. So, we use a commend line program to keep copying the data file and upload the copied file.

Moreover, if the data file become too large, the uploading process will be very slow. So, we clear the data file every midnight after the uploading.

(3). Data analysis programing

a. The elements controlled by user

There are 4 elements the user can control: number of detectors, gates, length of wires and the minimum number of detectors required to get a shower.

The gate is defined as the largest possible muon arrival time difference between different detectors, which means all the events happened in the gate time range might be caused by a shower. .Since the optical path difference is always less than the location difference of detector, the gate is determined by the formula:

$$gate = \frac{\Delta d}{c}$$

where c is the light speed.

There are some background noise on our detector. After the discriminator circuit, it will have frequency like 40 Hz. So, the fails coincidence rate (the coincidence caused by the noise which looks like a shower) is calculated by:

$$rate = C * (f)^n * gate^{n-1}$$

Where f is the background noise rate, n is the number of detectors and C is a constant related to n.

This rate should be low enough. That's why we may want the minimum number of detectors required to get a shower.

The length of wire will cause the time delay. Both of the wire of antenna and the wire connect the GPS receiver and the Arduino matter. They will make the measured time is earlier than actually time. So we need to add the delay to our data.

Since the electric signal travels in the light speed. The time delay should calculated by the formula:

$$delay = \frac{\Delta l}{c}$$

Moreover, not every detector will receive muon signal will there is a shower. So, there should be a minimum number of detectors required to get a shower.

b. The idea of the programing

We write 3 functions for preparation. First is the function to read data line by line and save them in an array. The second is to find the minimum time data from the array. The last is find the coincidence shower (time difference in the gate) about the minimum data in the array and read the next lane of the coincidence data.

Since the data files are sorted increasingly, we just the first line of all the files and they will be the minimum data of the files they are in.

Then, we use the second function to find the minimum data of the array and it will be the minimum of all the data. Then, we can find the data which is less than the minimum + gate in the array because if those data is not in the gate, other bigger data should not in the gate as well. Then read the next line of the data in the gate (minimum data will always be in the gate) and find the minimum again until the end of the files.

Since the program will reads all the data files once and the time consumed by the minimum function is proportional to the number of files. So, the complexity (c) will be $O(Nn^2)$, where N is the length of data files and n is number of data files.

```
at time 1.611694, have 2 detector receive signal, they are:
 4 5

at time 1.613939, have 3 detector receive signal, they are:
 1 4 5

at time 1.616185, have 3 detector receive signal, they are:
 1 4 5

at time 1.618429, have 4 detector receive signal, they are:
 1 2 4 5

at time 1.620675, have 4 detector receive signal, they are:
 1 2 4 5

at time 1.622919, have 4 detector receive signal, they are:
 1 2 4 5
```

This is a sample output of the data file analysis program which shows the time of shower and which detectors got coincidence.

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.

(4). Impact on Laboratory or National Missions.

The project will find the cosmic ray with different energy level. Some cosmic rays have a kinetic energy greater than1×10^{18} eV that people never got even with the best accelerators. So, we may have chance to find one a high energy cosmic ray and see what will happen when a high energy cosmic ray comes.

We also want to know where the cosmic ray comes from by the reconstruction. However, due to the earth magnet field affect, we can't the direction of the shower will be the direction of the cosmic ray coming from. More math work needs to be down with the earth magnet field graph.