

小型ラドン検出器の性能評価 及び改良

奈良女子大学

B4青山美嶺 西川愛

目次

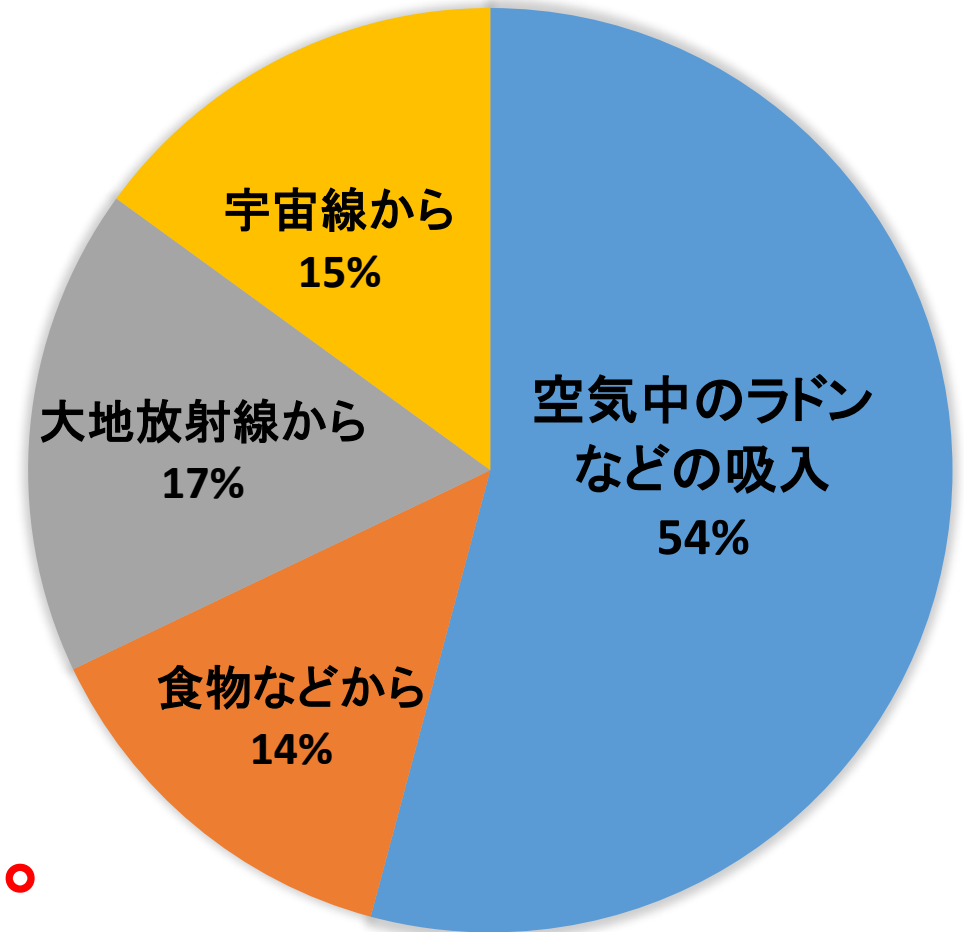
- ラドンについて
- 研究の目的
- 検出原理
- 実験 ①ラドンガスの測定
②Am線源の測定
- 半減期の導出
- 先行実験との比較
- まとめ
- 今後の展望

ラドンとは？

- ^{222}Rn 原子番号は86
- 無味無臭、無色の気体
- 年間被ばく量の約半分がラドンによるもの
- ラドン温泉

ラドンは身近な放射性物質である。

日本人の年間被ばく量2.4mSv



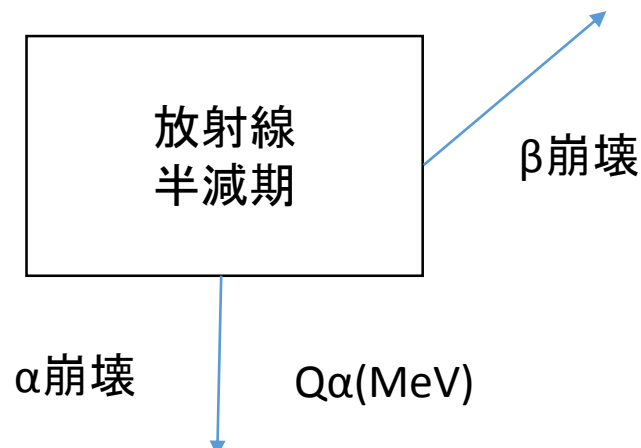
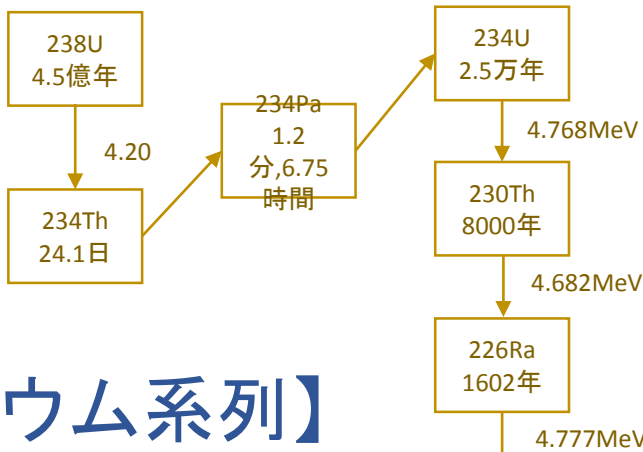
崩壊系列

存在比

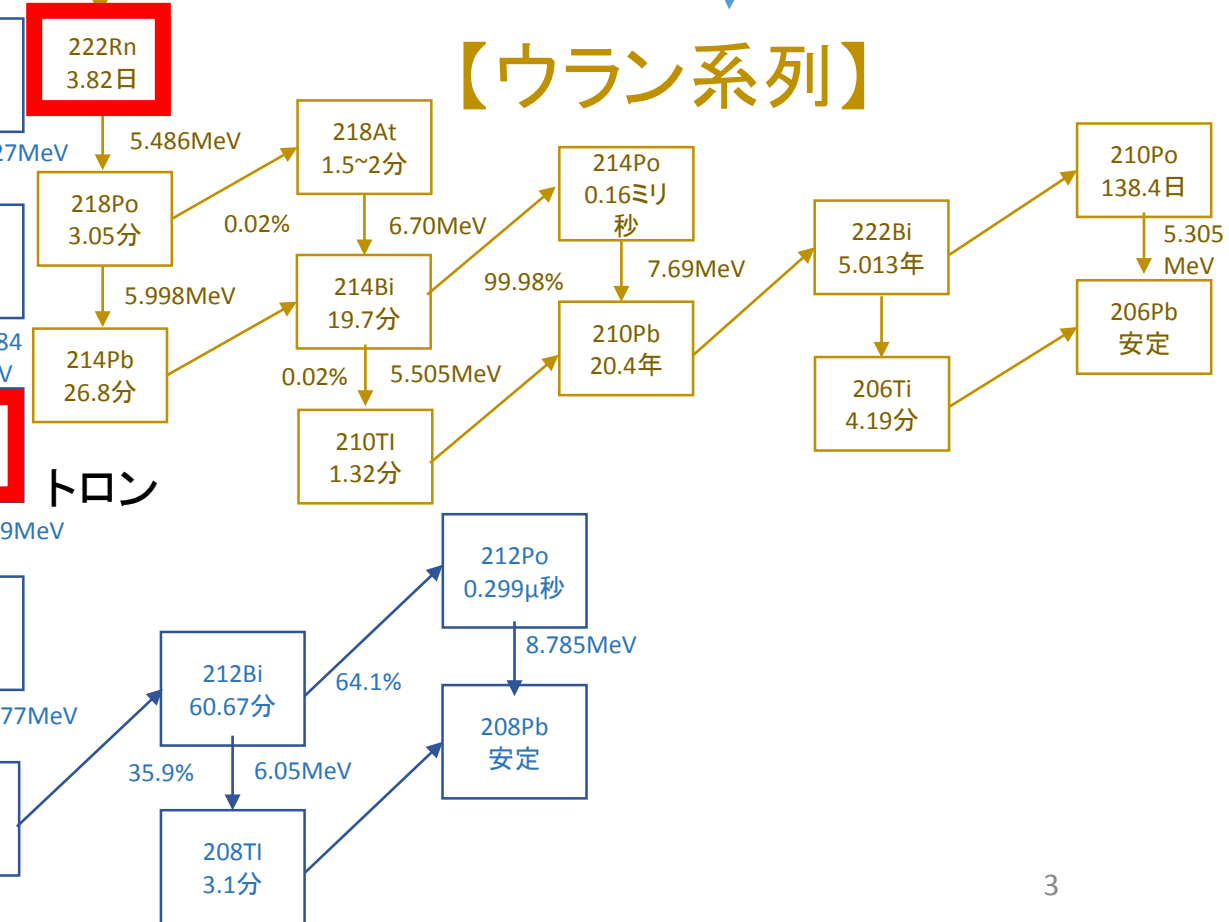
1:9:90

アクチニウム系列はほとんど存在しない

【トリウム系列】



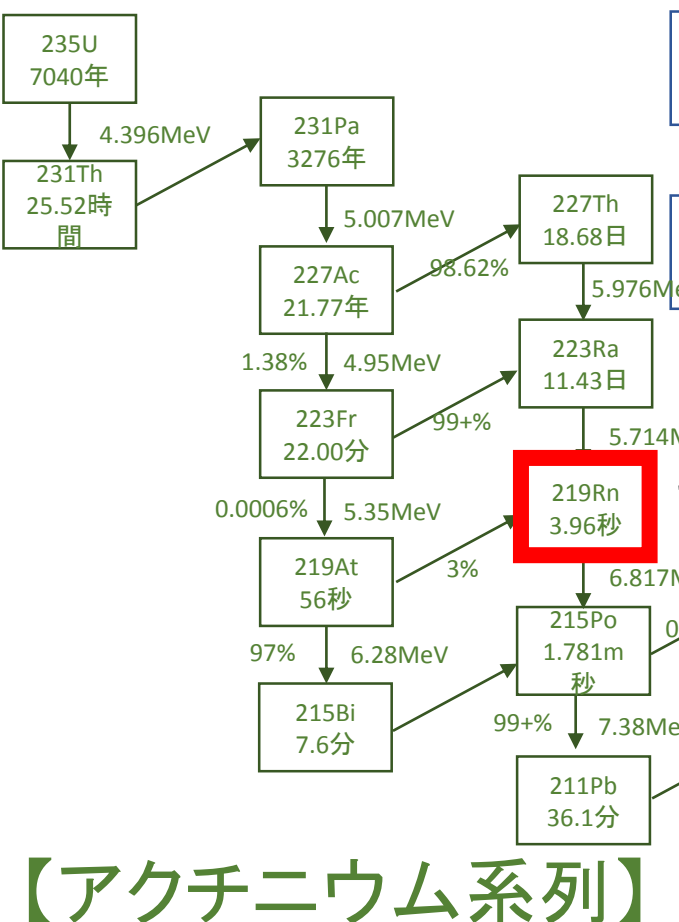
【ウラン系列】



アクチノン

トロン

【アクチニウム系列】



研究目的

- 小型化、安価で入手できるようなラドン検出器を開発
- ラドン検出器の性能評価と改良

目次

- ラドンについて
- 研究の目的
- 検出原理
- 実験 ①ラドンガスの測定
②Am線源の測定
- 半減期の導出
- 先行実験との比較
- まとめ
- 今後の展望

ラドン検出の全体部

ラドン検出器
+ PD

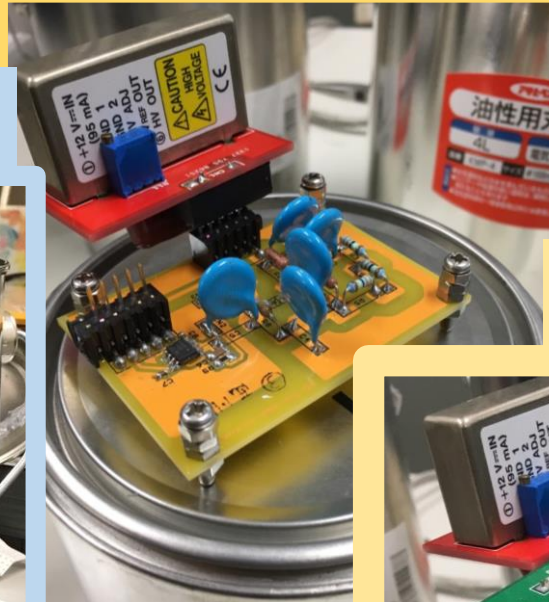
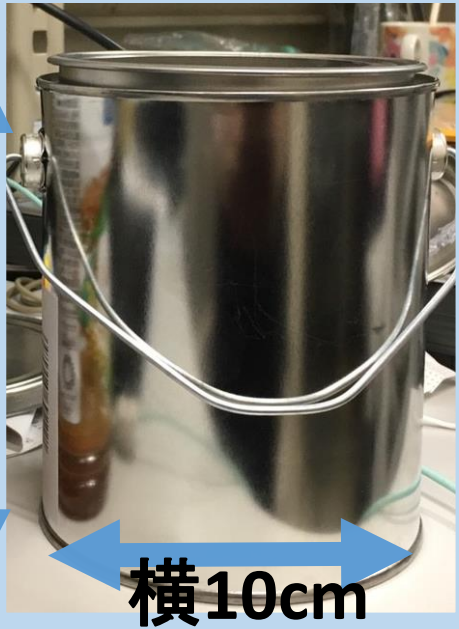
プリアンプ

シェイパー
アンプ

Arduino

PC

ラドン検出部



アナログ部



データ制御部

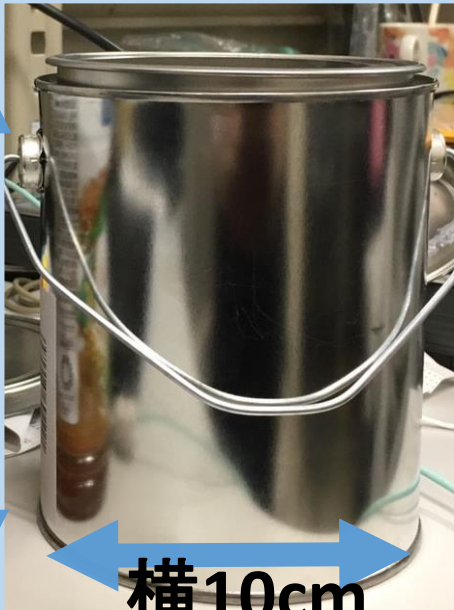


データ収集部



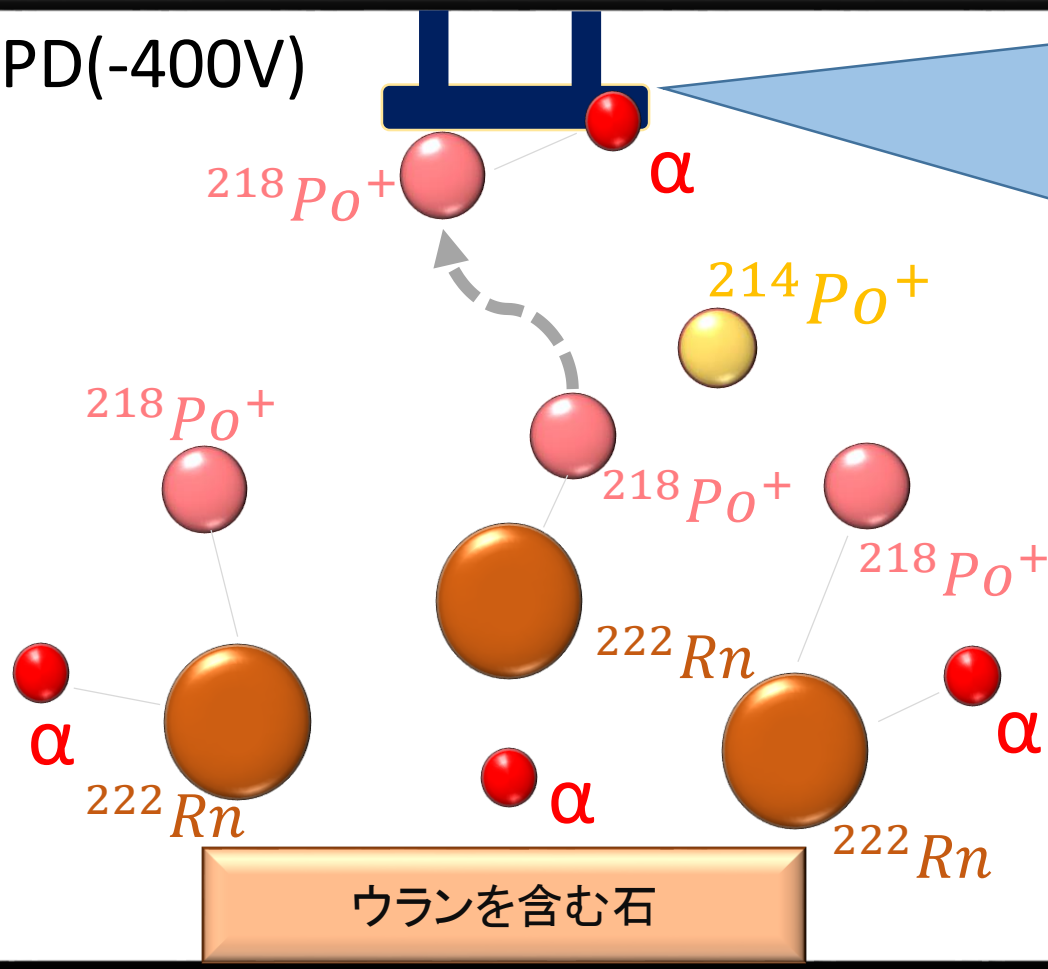
ラドン検出器
+PD

ラドン検出部



横1cm,縦1cm

PD(-400V)



5.998MeVの α 線が入射することによって、 2.7×10^{-13} (C)の電荷を作る。

静電捕集法
PD表面に負の高電圧をかけ、缶をGNDにし、缶内部に静電場を作ることによって、陽イオンになっているPoをPD表面に効率よく集める。

0V

ラドン検出器
+PD

プリアンプ

シェイパー
アンプ

Ardu

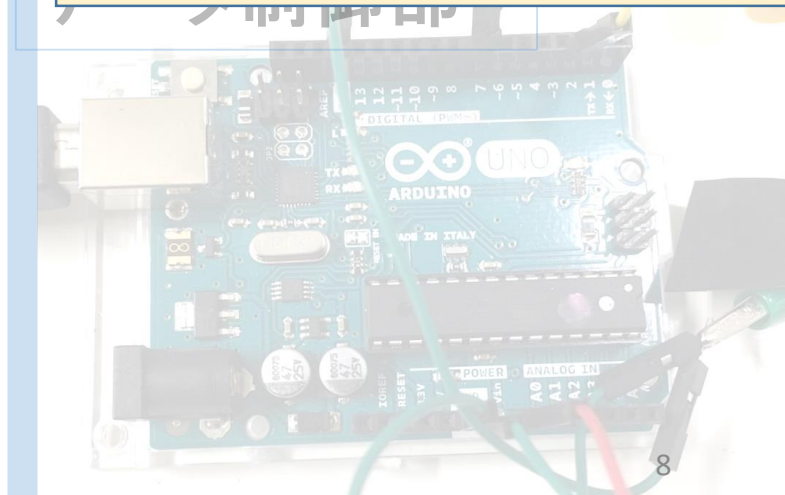
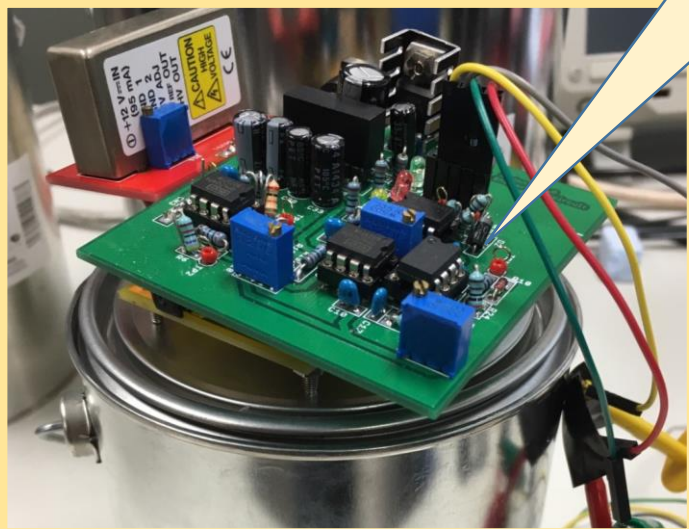
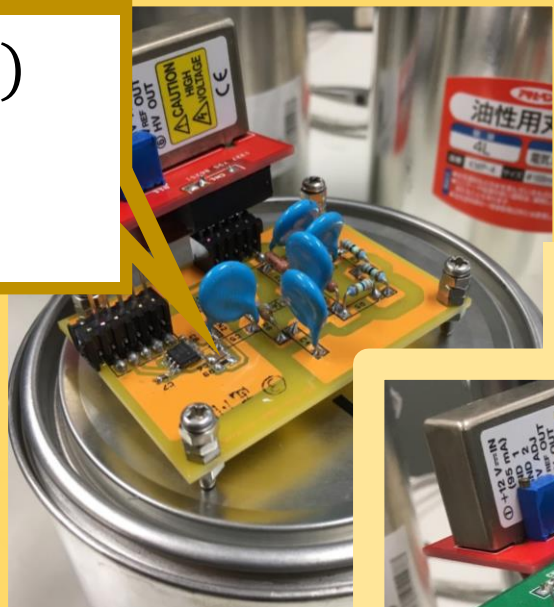
$2.7 \times 10^{-13} (C)$
の電荷を27mVの
電圧に変える。

アナログ部

プリアンプからきた
電圧値を増幅し、
ノイズ軽減も行う。
プリアンプからきた
信号を約100倍にして
Arduinoに出力。

縦
13
cm

横10cm



ラドン検出器
+PD

プリアンプ

シェイパー
アンプ

Arduino

PC

Arduinoはハードウェア、ソフトウェアともにオープンソースである。
デジタル・アナログ信号の入出力が可能であり、0V~5Vの電圧値を1024chに変換する。
2.7Vは約580chに変換され、PCへ。

データ収集部

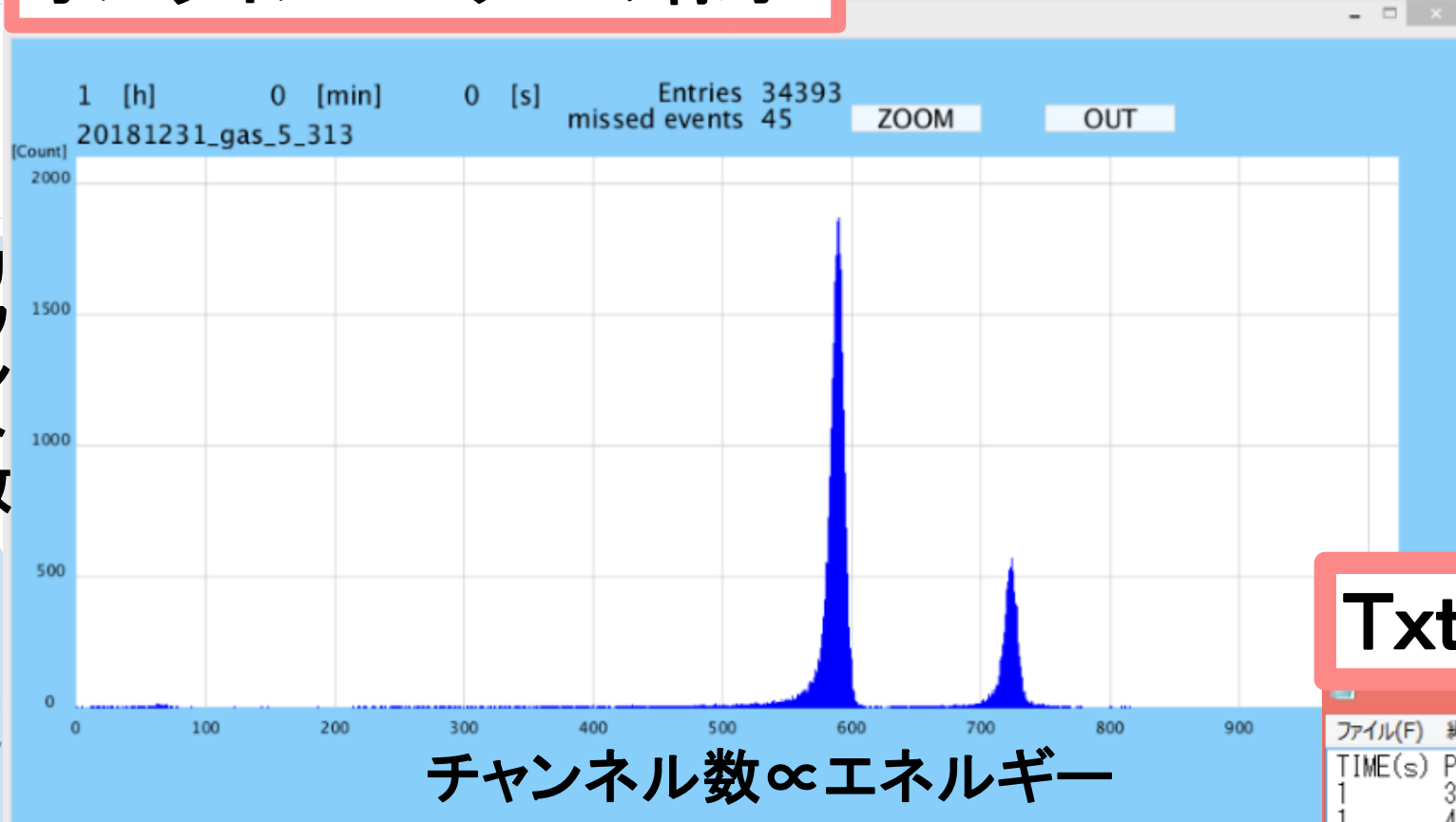
データ制御部



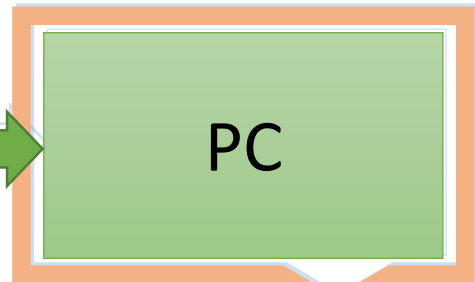
横10cm

縦13cm

オンラインモニターの様子



Arduino



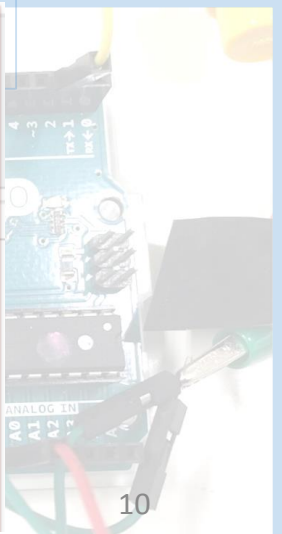
PC



データ収集部

Txtファイル

TIME(s)	PulseHeight (ch)
33	33
46	-3
76	76
92	90
90	67
50	50
40	28
38	38
55	55
55	55
36	36
41	41



縦
13
cm

横
10
cm

目次

- ラドンについて
- 研究の目的
- 検出原理
- 実験 ①ラドンガスの測定
②Am線源の測定
- 半減期の導出
- 先行実験との比較
- まとめ
- 今後の展望

コンセント

Arduino



PC



シェイパーアンプ



HV



プリアンプ



暗幕

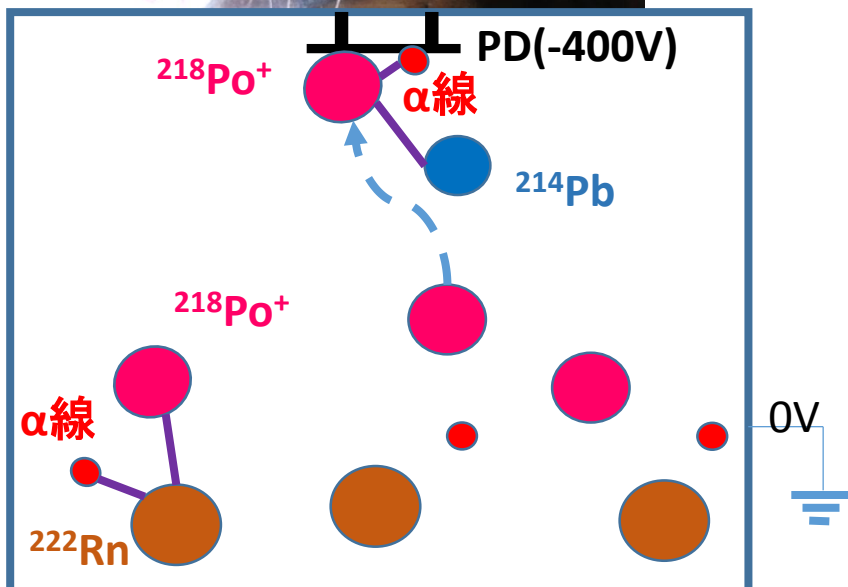
ラドンガスの測定

DOLL STONEを検出器に入れてラドンガスを充満させ、測定開始時にDOLL STONEを取り除いて測定した。



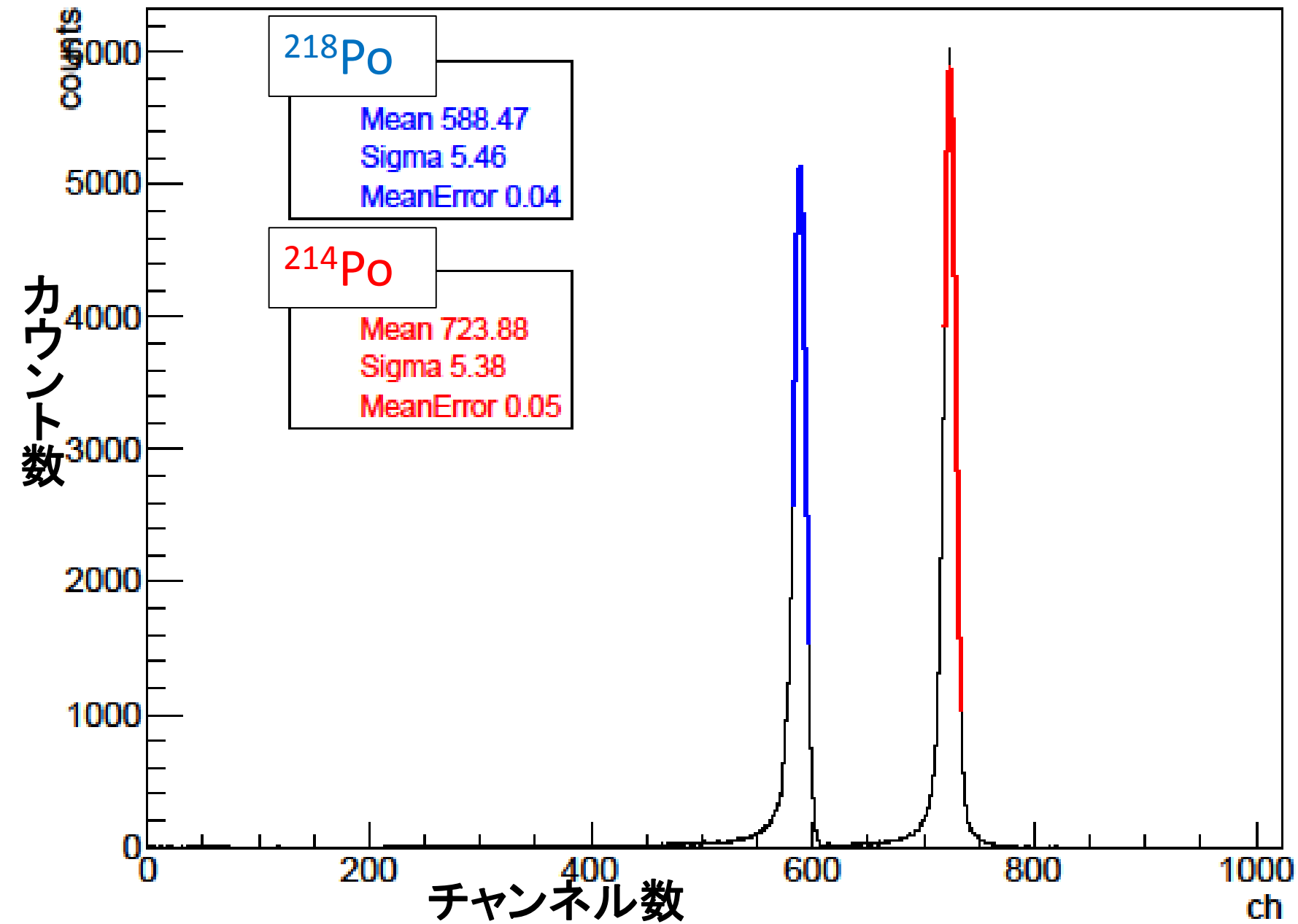
DOLL STONE

天然のウランが少し含まれている

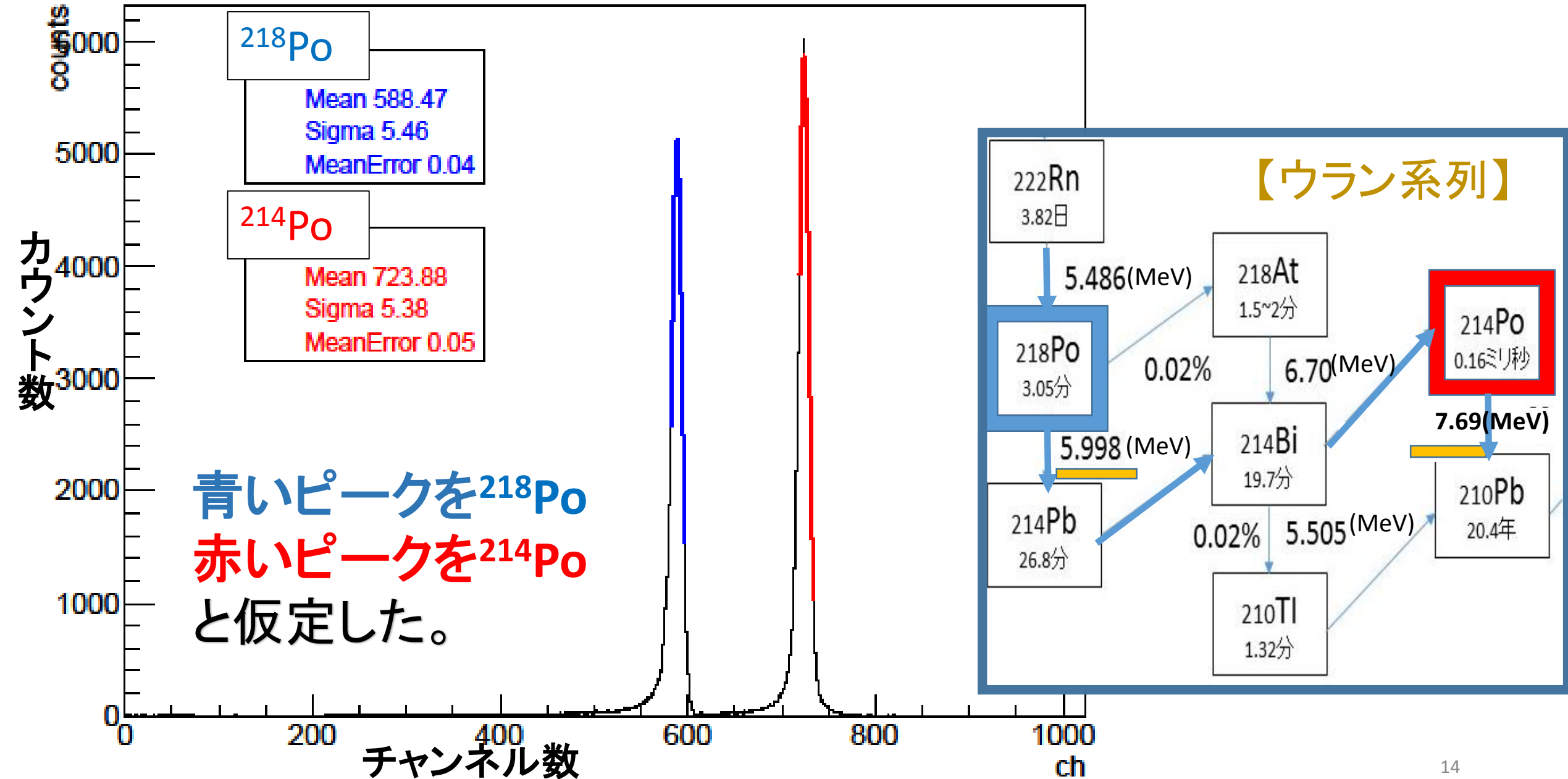


→ラドン検出器の全体写真

ラドンガスの波高分布

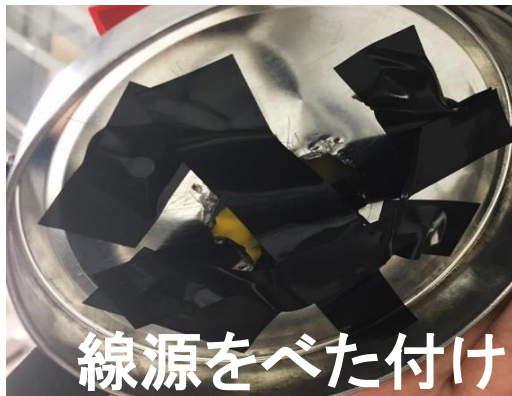
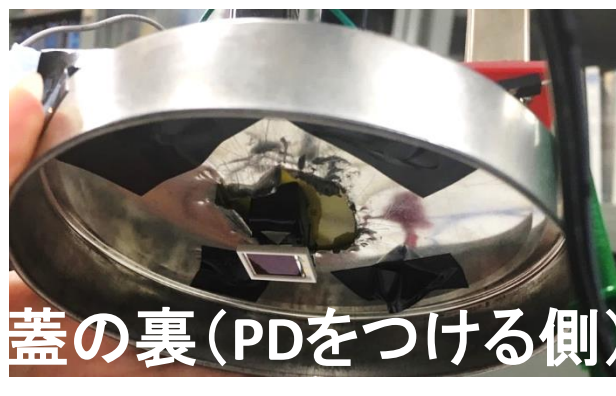


ラドンガスの波高分布



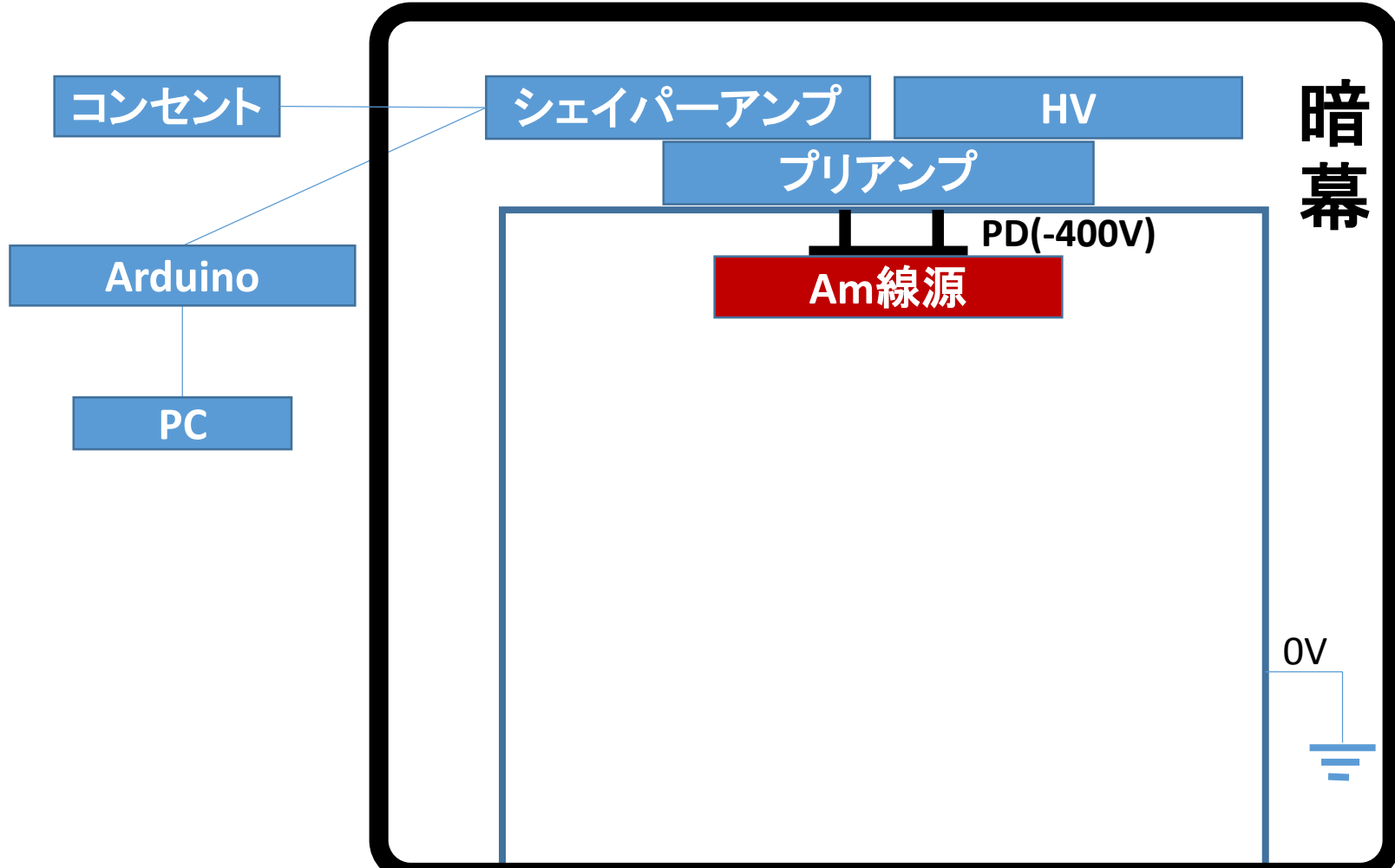
目次

- ラドンについて
- 研究の目的
- 検出原理
- 実験 ①ラドンガスの測定
②Am線源の測定
- 半減期の導出
- 先行実験との比較
- まとめ
- 今後の展望



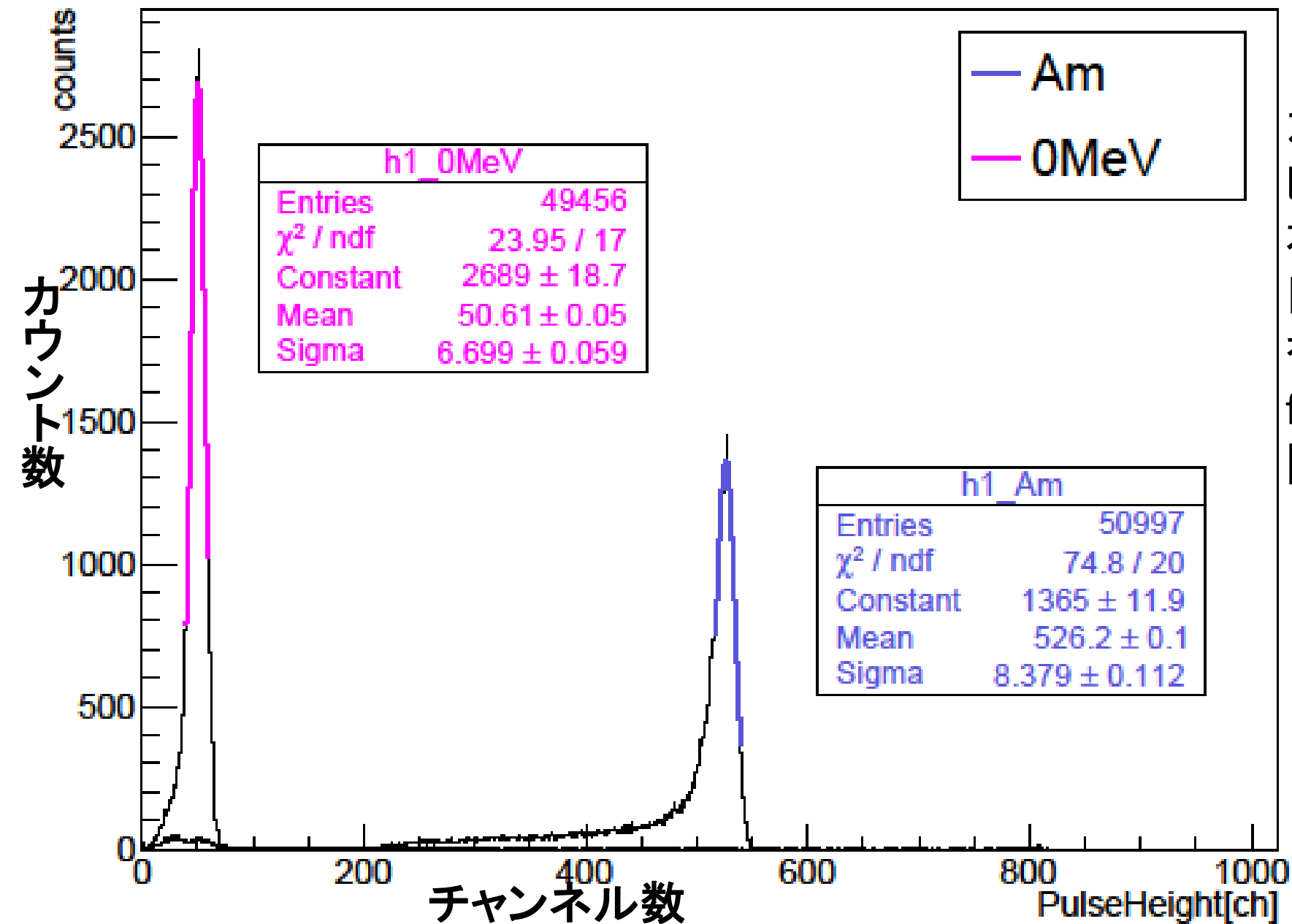
Am線源の測定

chとエネルギーの関係を調べるため、線源を用いて0MeV時と線源から出る α 線のchを測定。今回使用した ^{241}Am 線源の α 線のエネルギーは密封線源の窓の損失分を考えて5.3MeVであるとした。



使用した ^{241}Am 線源

Am線源測定時の波高分布

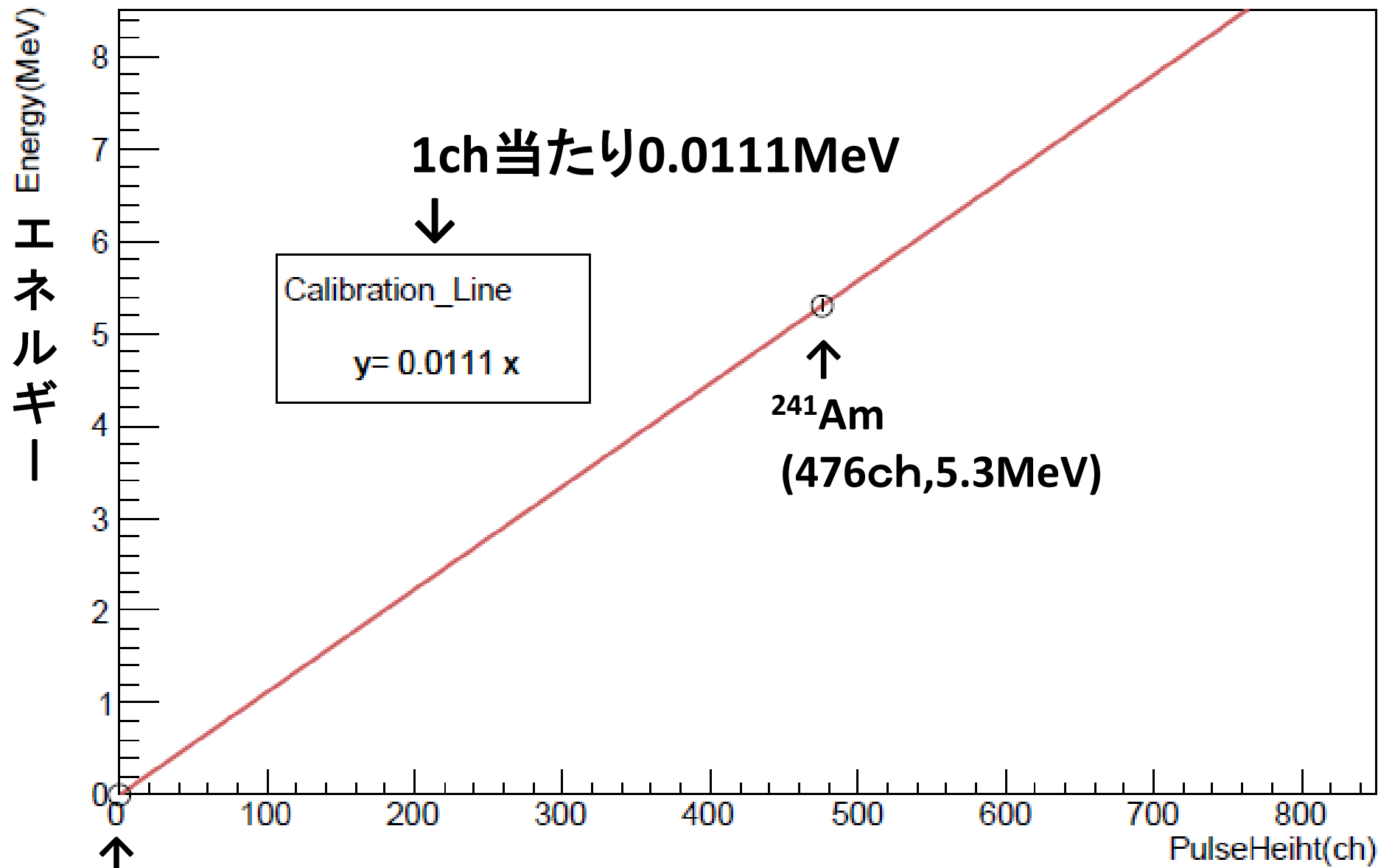


左側のピンク色でfitしているヒストグラムが0MeVの、右側の青色でfitしているヒストグラムがAmのヒストグラムを表している。

fitting 結果とエネルギーの関係は以下のようなになる。

	0MeV	Am
エネルギー [MeV]	0.0	5.3
対応するch	50.61	526.2
対応するchの誤差	0.05	0.1

Am校正直線との比較

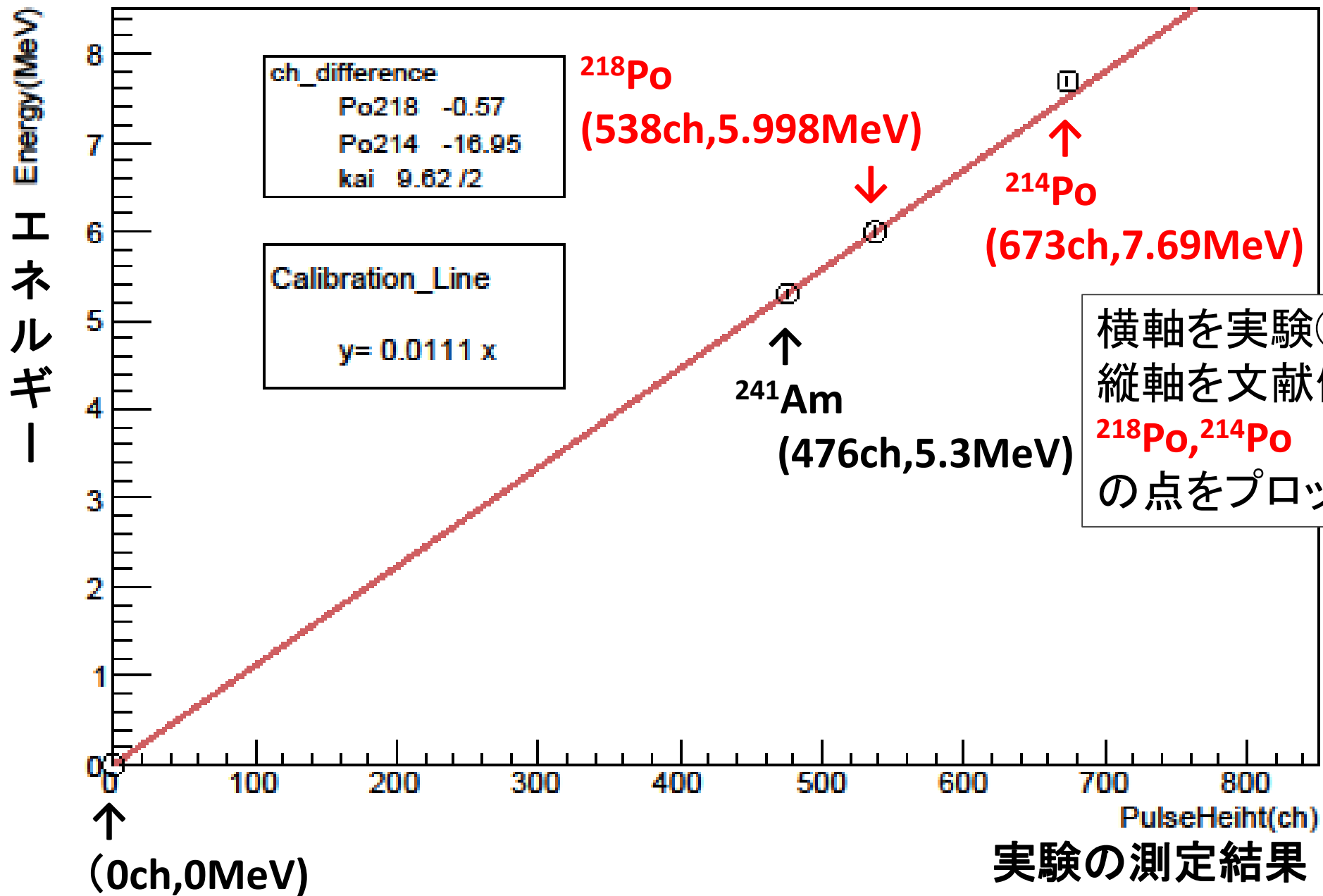


二点を用い
校正直線を引いた。

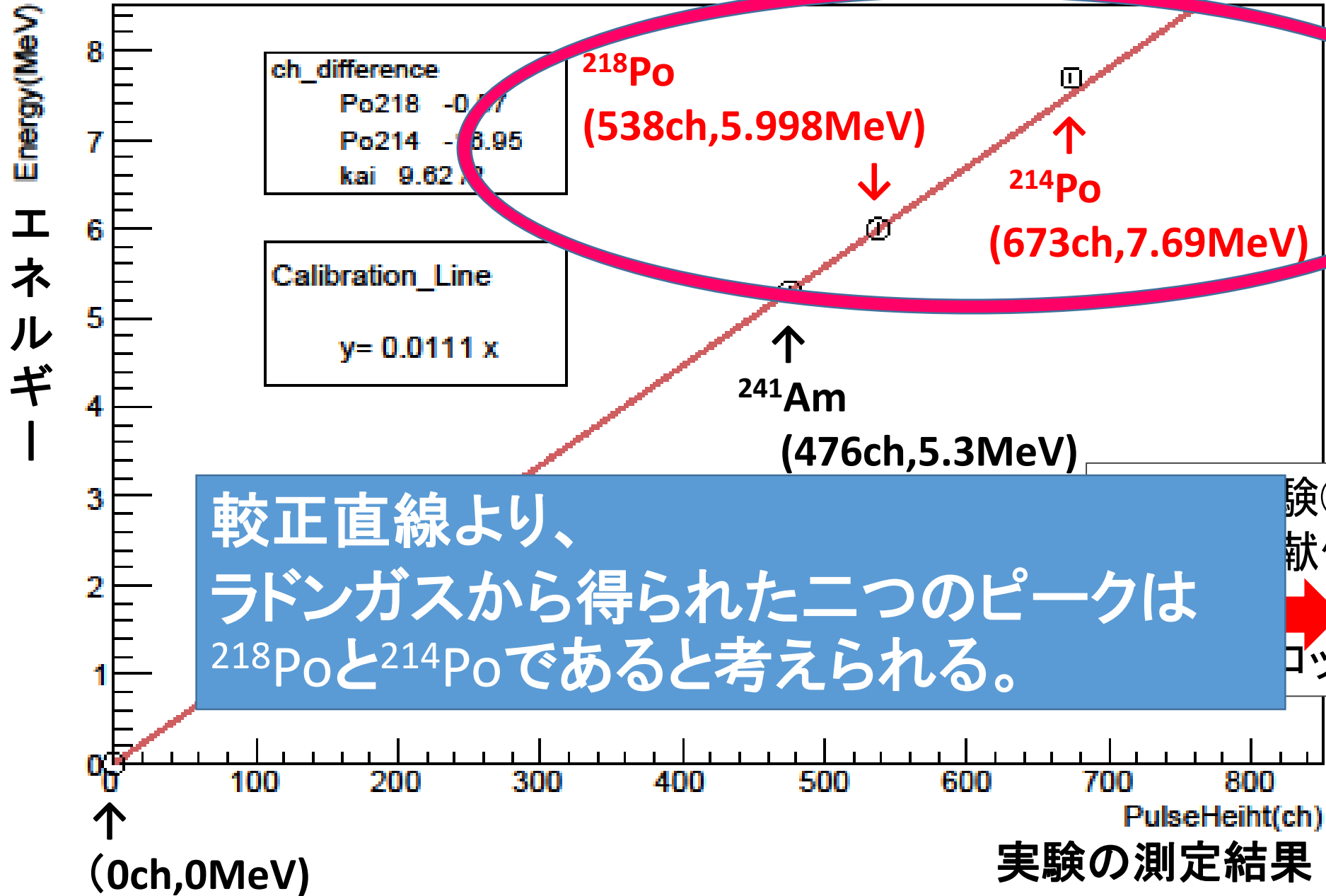
↑
(0ch,0MeV)

実験の測定結果

Am校正直線との比較



Am校正直線との比較



校正直線より、ラドンガスから得られた二つのピークは ^{218}Po と ^{214}Po であると考えられる。

次はそれらの半減期を求め
る。

目次

- ラドンについて
- 研究の目的
- 検出原理
- 実験 ①ラドンガスの測定
②Am線源の測定
- **半減期の導出**
- 先行実験との比較
- まとめ
- 今後の展望

崩壊定数

崩壊定数とは単位時間あたりに放射性原子核が崩壊する確率のことであり、

微小時間 dt の間に崩壊する核の個数 $dN(t)$ は、崩壊定数 λ を用いて

$$dN(t) = -\lambda N(t)dt \quad \text{であり、}$$

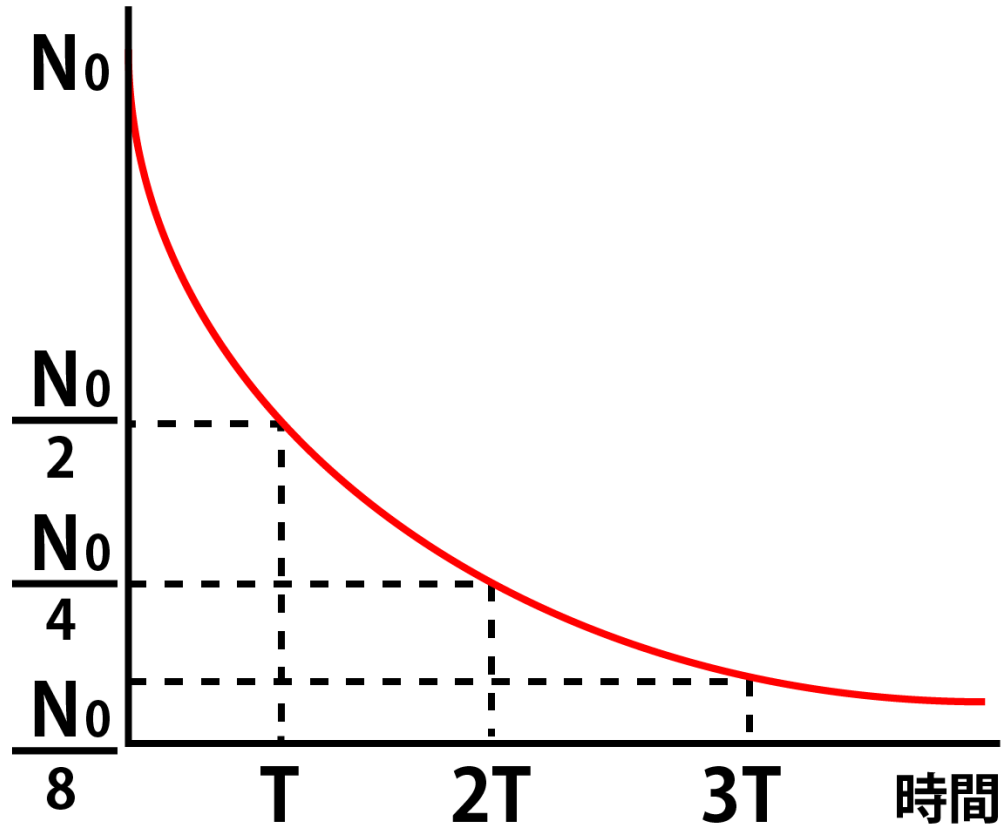
微分方程式を解くと

$$N(t) = N_0 e^{-\lambda t} \quad (2)$$

よって(1),(2)より $\lambda T = \ln 2$

→崩壊定数がわかれば半減期が求まる!!!

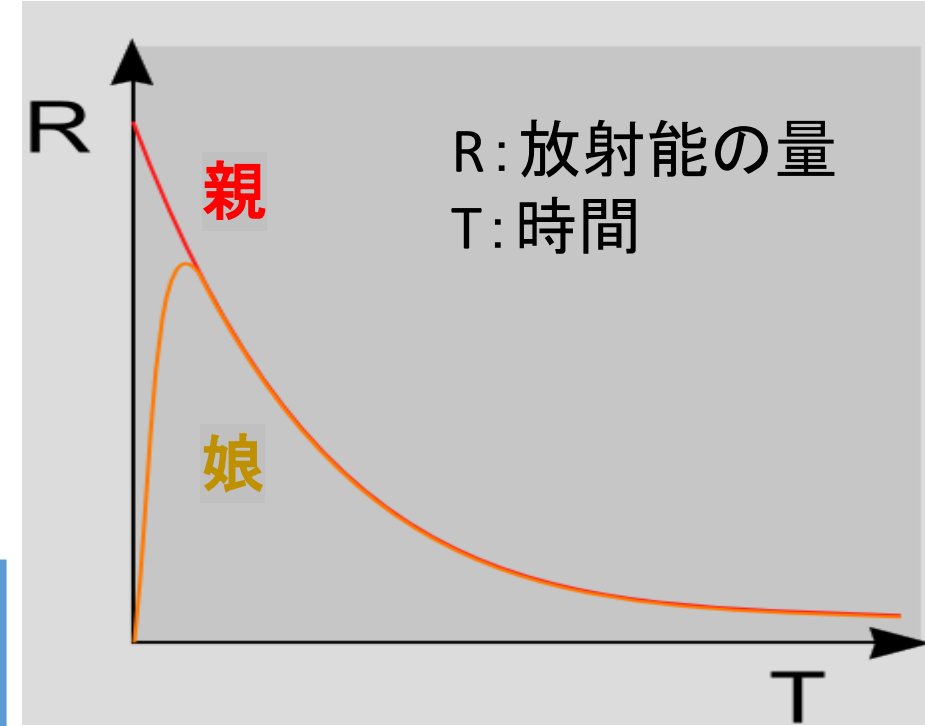
原子核の個数(残存量)



半減期が T の原子核が N_0 個あるとすると、

$$N(t) = N_0 \left(\frac{1}{2}\right)^{\frac{t}{T}} \quad (1)$$

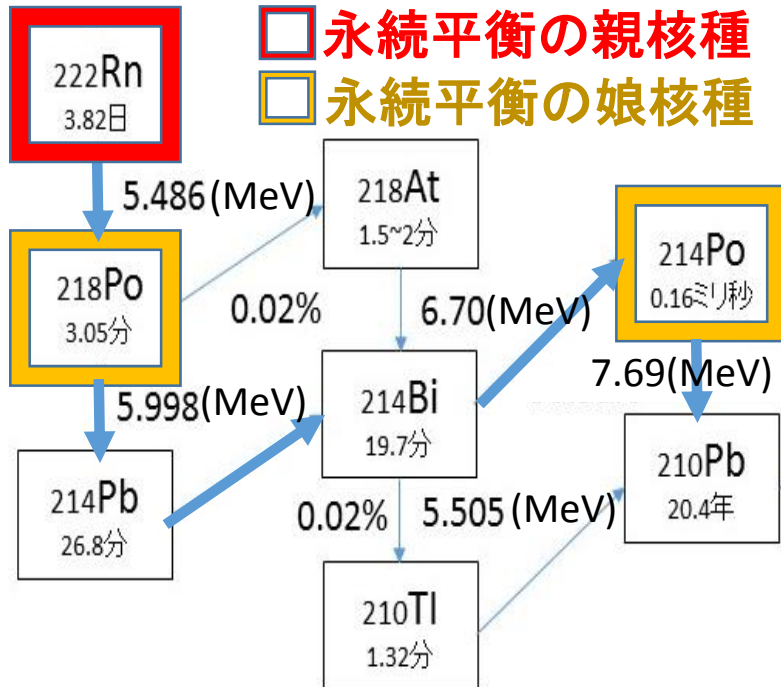
永続平衡



^{222}Rn と二つのPoは永続平衡の関係になっている！！

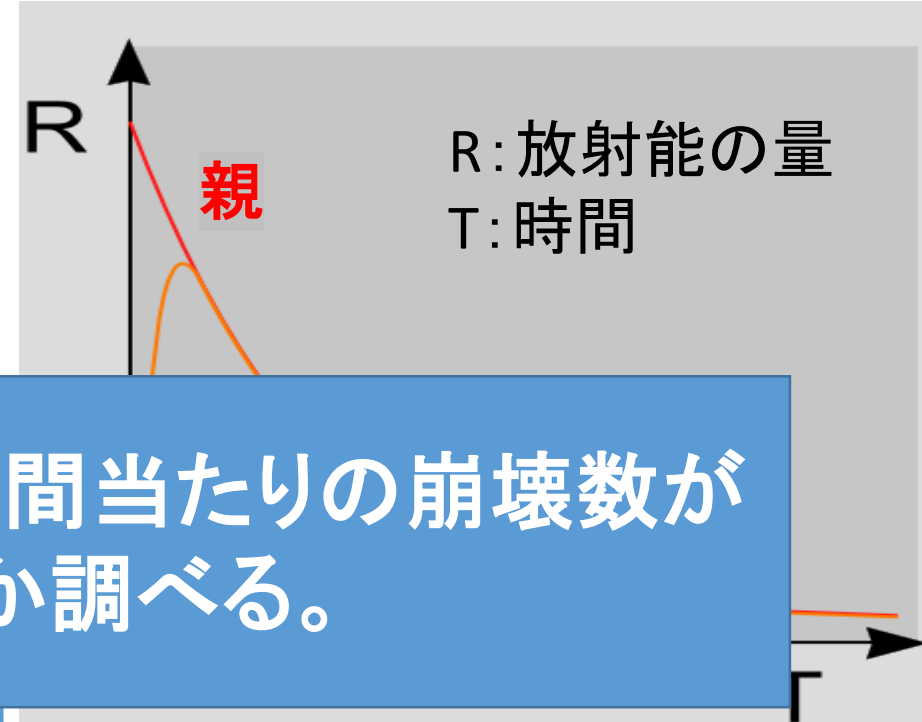
親核種が娘核種より半減期が非常に長ければ...

親核種の崩壊が娘核種の量を決めるために、親核種の放射エネルギーと娘核種の放射エネルギーは等しくなり、親核種の半減期カーブに沿って時間とともに減衰していく。このことを**永続平衡**という。



名前	T (半減期)
^{222}Rn (親)	3.82日
^{218}Po (娘)	3.05分
^{214}Po (娘)	1.6×10^{-4} 秒

永続平衡

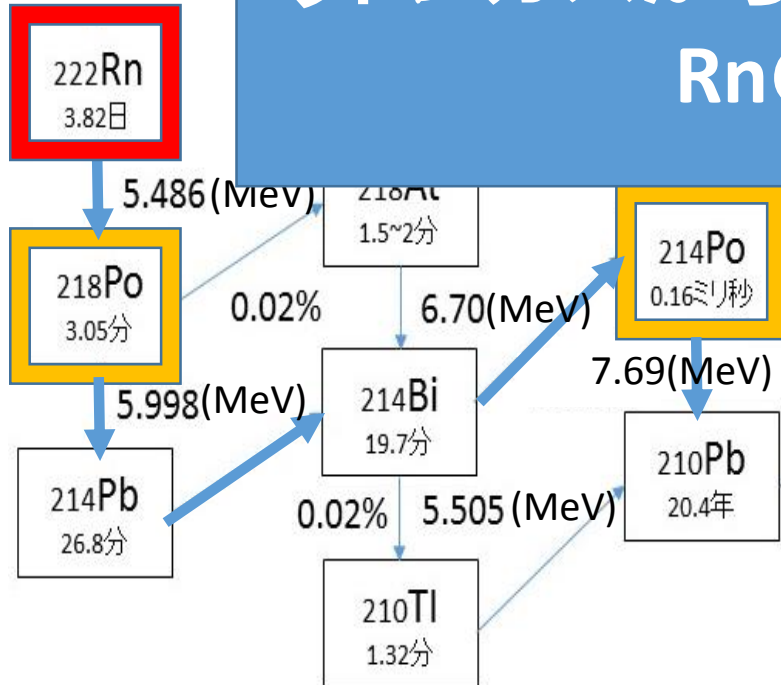


親核種が娘核種より半減期が非常に長ければ...

親核種の崩壊が娘核種の量を決めるために、親核種の放射エネルギーと娘核種の放射エネルギーは等しくなり、親核種の半減期カーブに沿って時間とともに減衰していく。

このことを

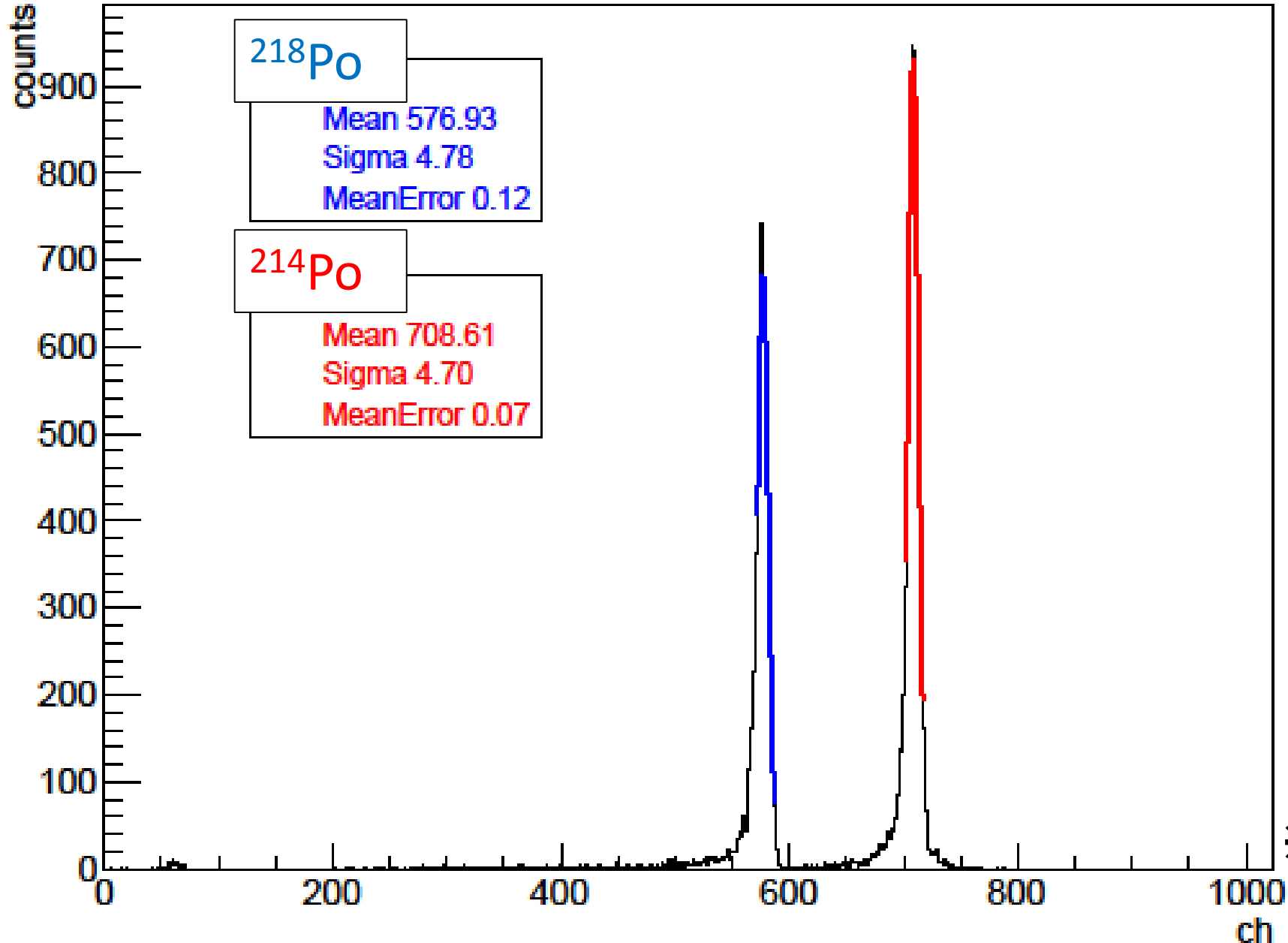
ラドンガスから測定したPoの単位時間当たりの崩壊数がRnの半減期と一致するか調べる。



222Rn (親)	3.82日
218Po (娘)	3.05分
214Po (娘)	1.6×10^{-4} 秒

222Rnと二つのPoは永続平衡の関係になっている！！

ラドンガスの波高分布



半減期測定

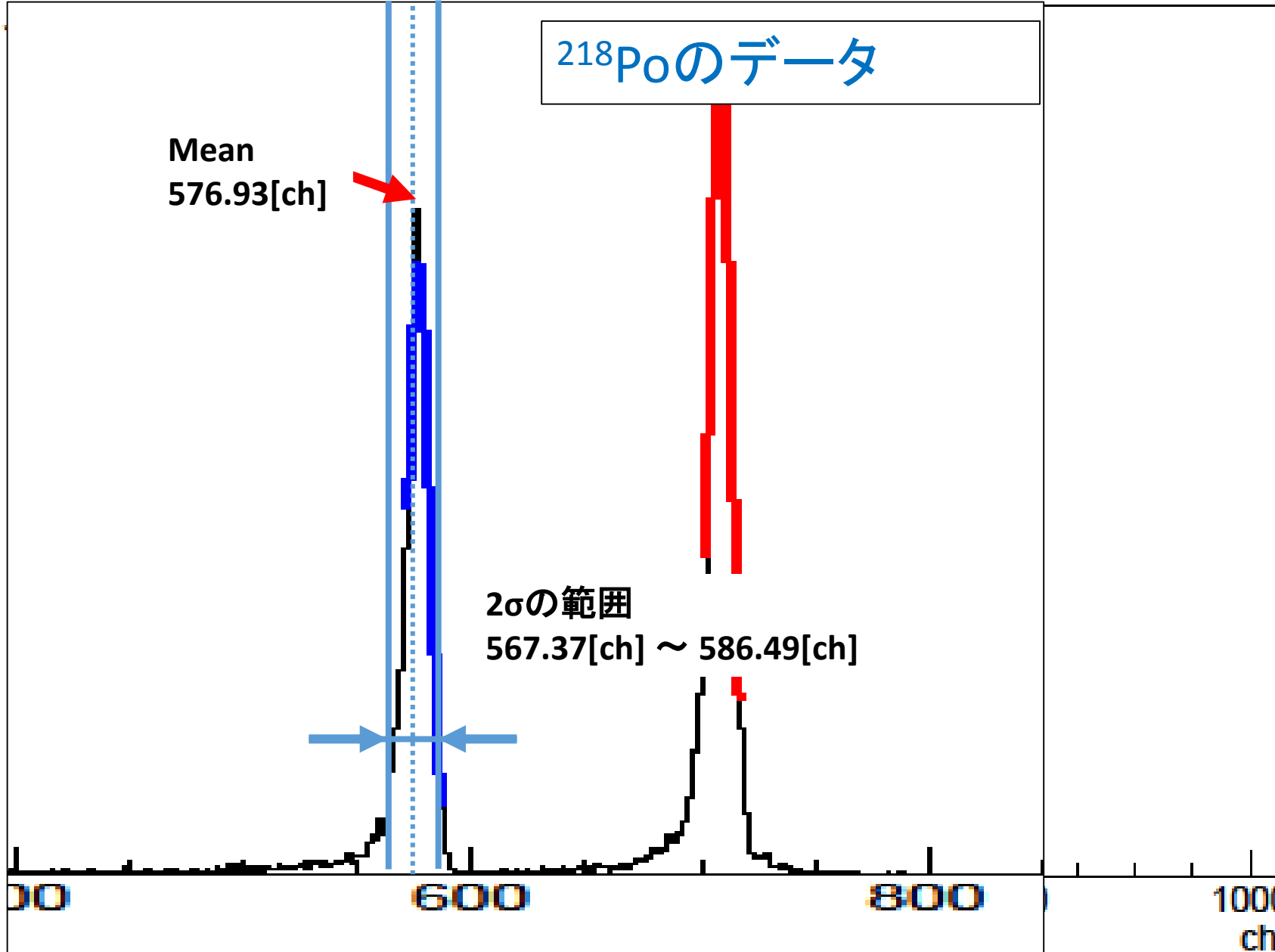
^{218}Po , ^{214}Po のそれぞれの単位時間当たりの崩壊数を調べるために、

ピークから 2σ の範囲に入っているデータをそれぞれ ^{218}Po , ^{214}Po として、単位時間ごとにいくつ検出しているかを調べる。

	Mean	σ
^{218}Po	576.93	4.78
^{214}Po	708.61	4.70

※先行実験検出器での測定データ

ラドンガスの波高分布



半減期測定

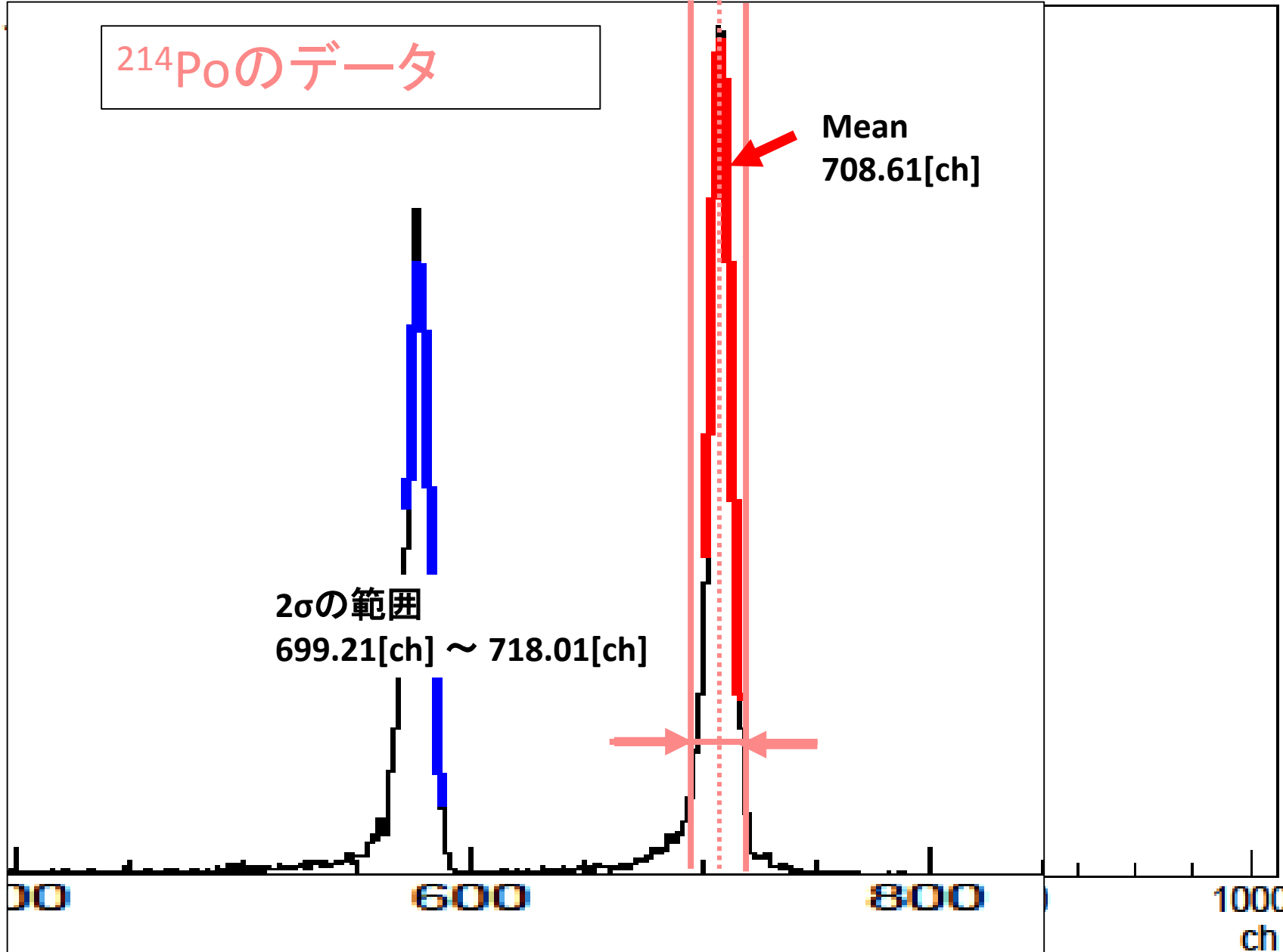
^{218}Po , ^{214}Po のそれぞれの単位時間当たりの崩壊数を調べるために、

ピークから 2σ の範囲に入っているデータをそれぞれ ^{218}Po , ^{214}Po として、単位時間ごとにいくつ検出しているかを調べる。

	Mean	σ
^{218}Po	576.93	4.78
^{214}Po	708.61	4.70

※先行実験検出器での測定データ

ラドンガスの波高分布



半減期測定

^{218}Po , ^{214}Po のそれぞれの単位時間当たりの崩壊数を調べるために、

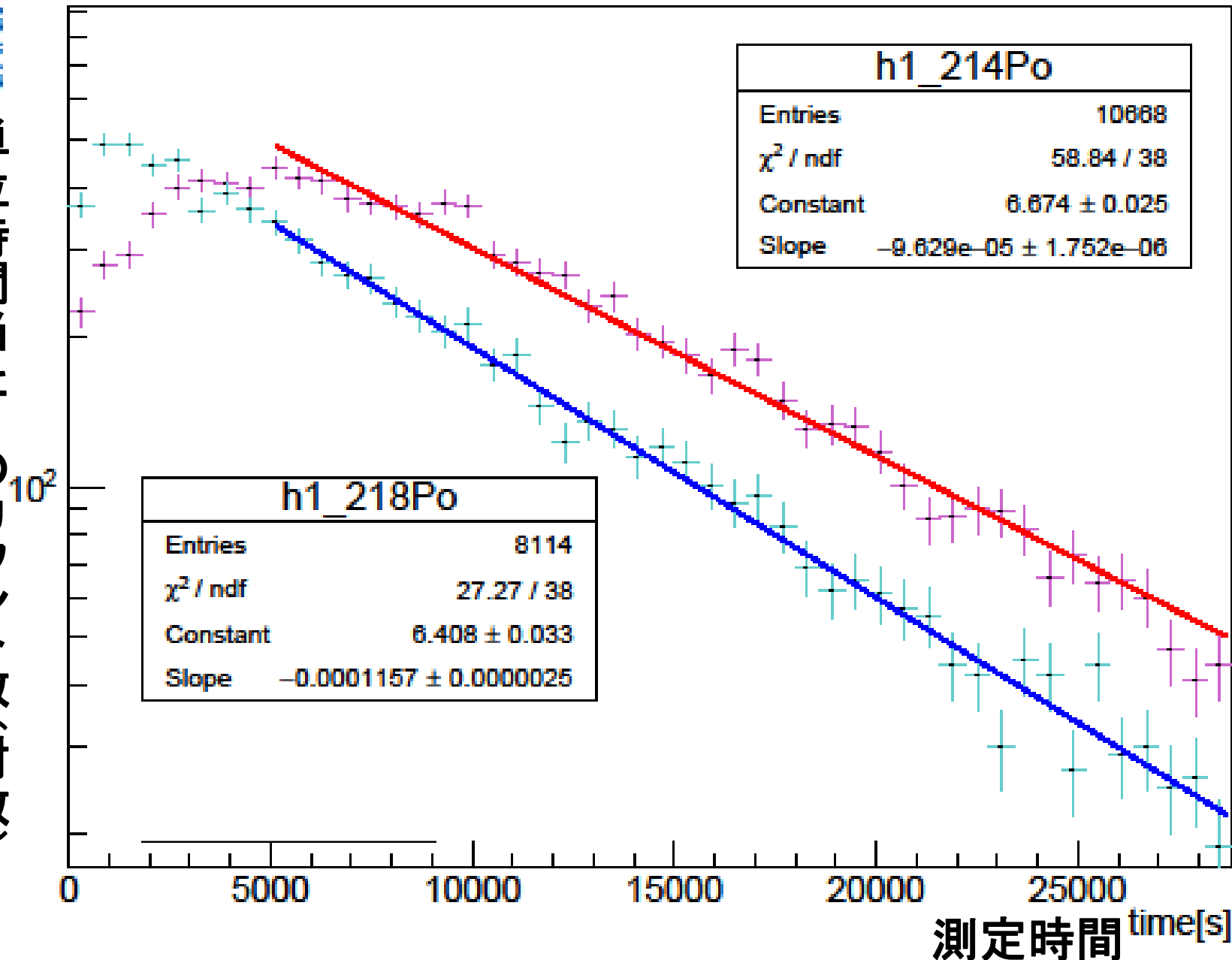
ピークから 2σ の範囲に入っているデータをそれぞれ ^{218}Po , ^{214}Po として、単位時間ごとにいくつ検出しているかを調べる。

	Mean	σ
^{218}Po	576.93	4.78
^{214}Po	708.61	4.70

※先行実験検出器での測定データ

Time-dN/dt

単位時間当たりのカウント数(対数)



半減期測定結果

指数関数でfitしており、

$$N(t) = N_0 e^{-\lambda t}$$

$$\log N(t) = \log N_0 - \lambda t$$

より、

fit線の傾きの大きさが崩壊定数 λ となり、

$\lambda T = \ln 2$ より

崩壊定数がわかれば半減期が求まる!!

	崩壊定数 λ [/s]	半減期 [day]
文献値 ^{222}Rn	2.09×10^{-6}	3.82
^{218}Po	1.157×10^{-4} $\pm 2.5 \times 10^{-6}$	$0.069 \pm$ 0.001
^{214}Po	9.629×10^{-5} $\pm 1.752 \times 10^{-6}$	$0.083 \pm$ 0.002

Time-dN/dt

半減期測定結果

指数関数でfitしており、

$$N(t) = N_0 e^{-\lambda t}$$

$$\log N(t) = \log N_0 - \lambda t$$

より、

fit線の傾きの大きさが

単位時間当たりのカウント数(対数)

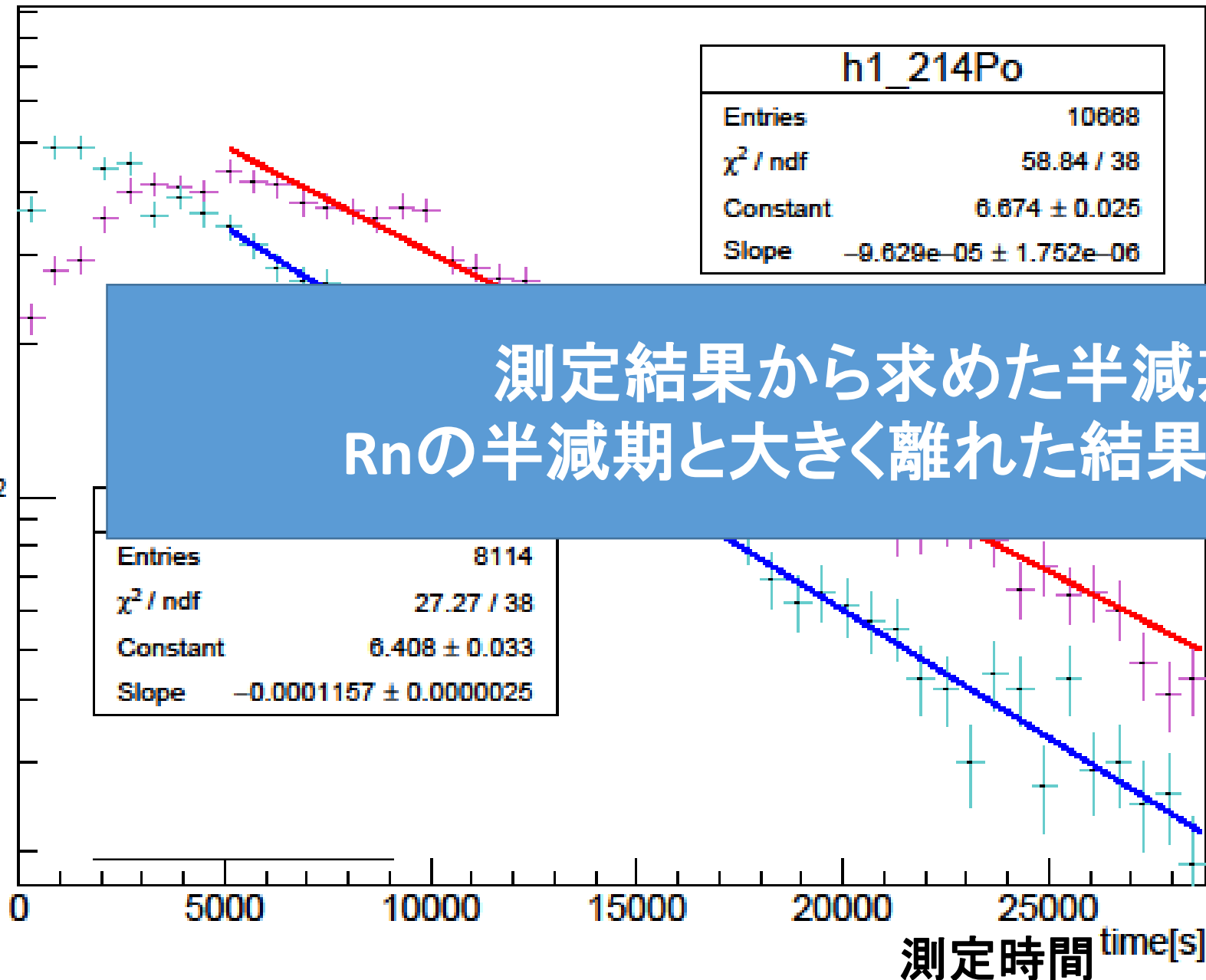
h1_214Po	
Entries	10888
χ^2 / ndf	58.84 / 38
Constant	6.674 ± 0.025
Slope	$-9.629e-05 \pm 1.752e-06$

測定結果から求めた半減期は Rnの半減期と大きく離れた結果となった。

半減

Entries	8114
χ^2 / ndf	27.27 / 38
Constant	6.408 ± 0.033
Slope	-0.0001157 ± 0.0000025

	崩壊定数 λ [/s]	半減期 [day]
文献値 ²²² Rn	2.09×10^{-6}	3.82
²¹⁸ Po	1.157×10^{-4} $\pm 2.5 \times 10^{-6}$	$0.069 \pm$ 0.001
²¹⁴ Po	9.629×10^{-5} $\pm 1.752 \times 10^{-6}$	$0.083 \pm$ 0.002



測定時間 time[s]

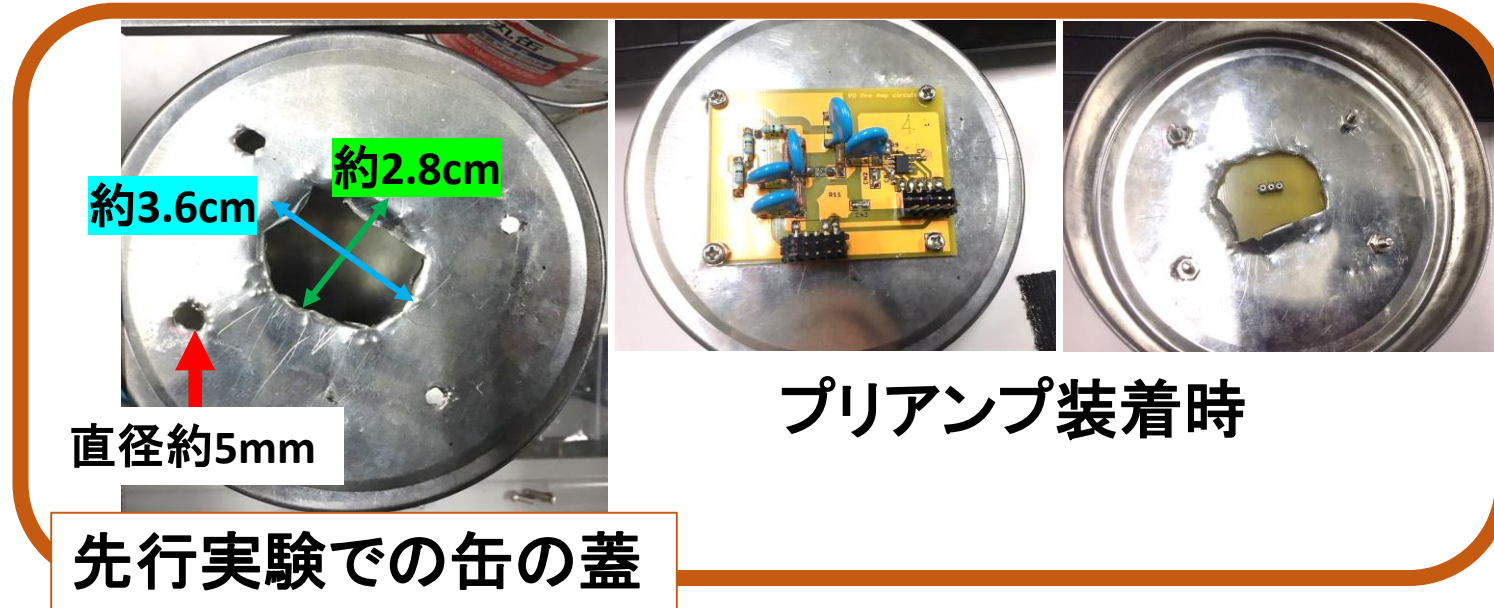
原因の考察と改良

原因の考察...

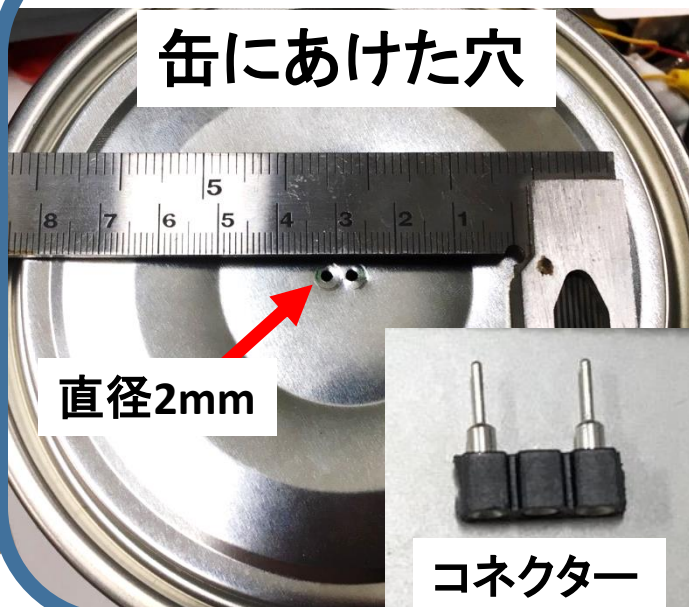
- ・缶の密封度
- ・缶の体積
- ・温度や湿度

改良...

まず缶の密封度を高める工夫をした。



改良した缶の蓋



最終的な完成図



プリアンプ装着時



目次

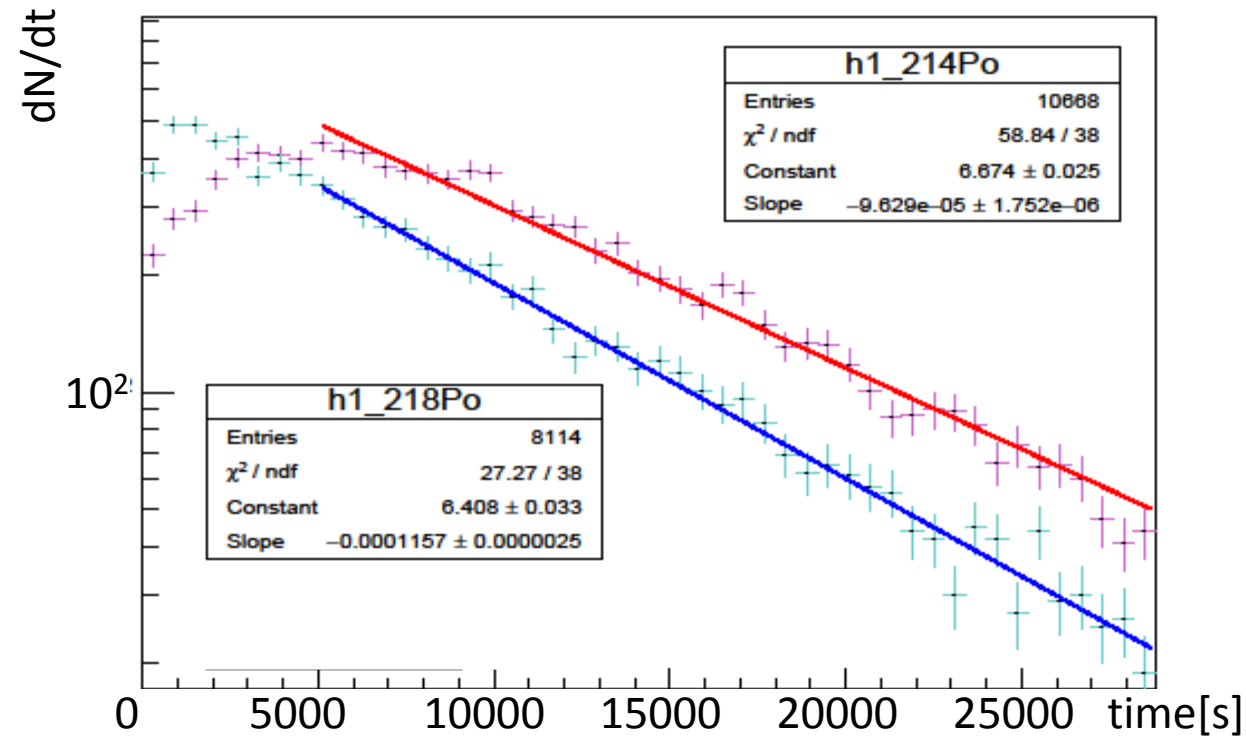
- ラドンについて
- 研究の目的
- 検出原理
- 実験 ①ラドンガスの測定
②Am線源の測定
- 先行実験との比較
- まとめ
- 今後の展望

先行実験との比較

左は先行実験で作成した小型ラドン検出器での測定結果、
右は体積がほぼ同じで密封度を高める工夫をした小型ラドン検出器での
測定結果である。(1bin=10分)

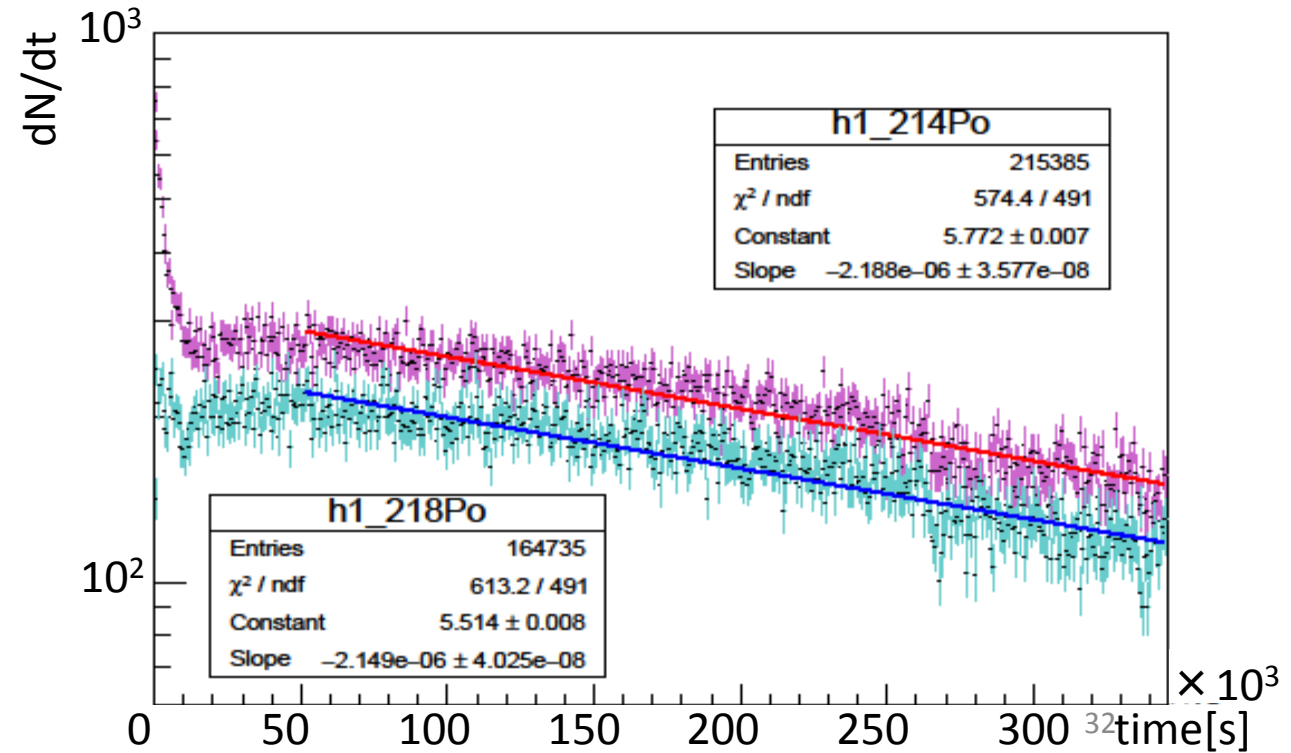
先行実験の小型ラドン検出器

Time-dN/dt



密封度を高めた小型ラドン検出器

Time-dN/dt

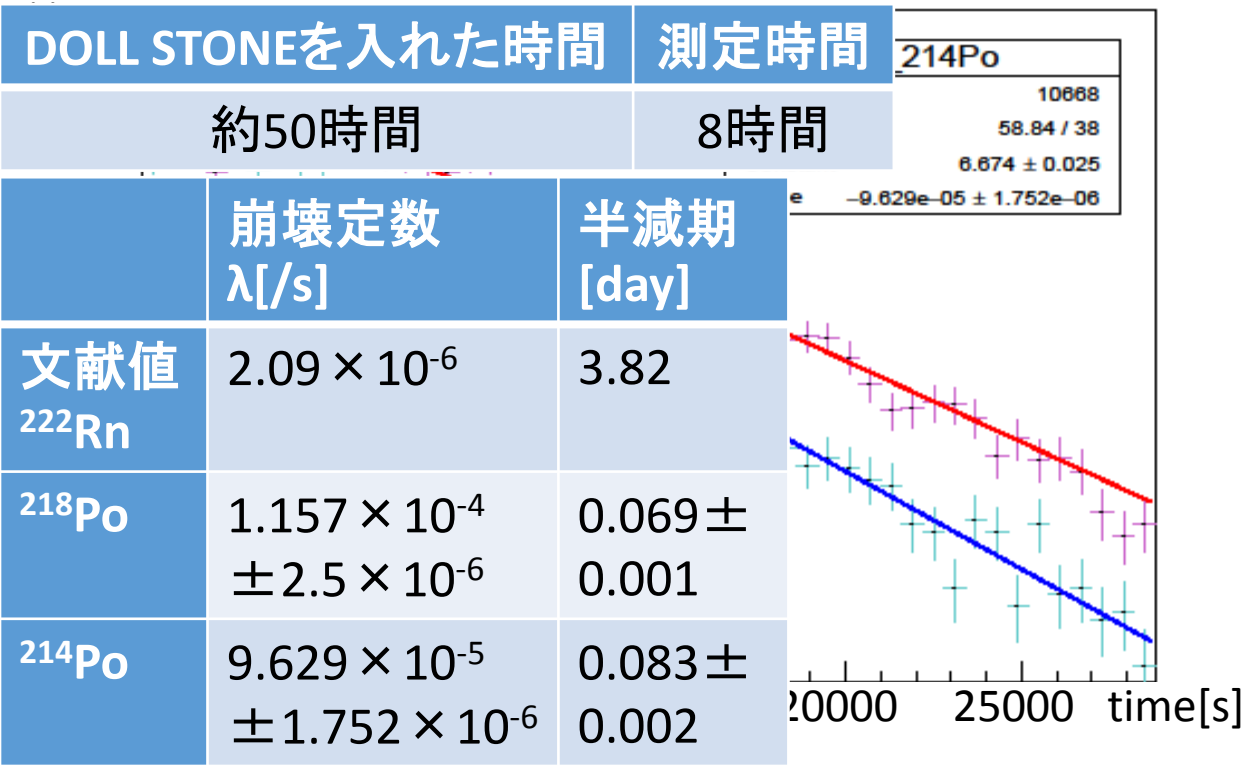


先行実験との比較

左は先行実験で作成した小型ラドン検出器での測定結果、右は体積がほぼ同じで密封度を高める工夫をした小型ラドン検出器での測定結果である。(1bin=10分)

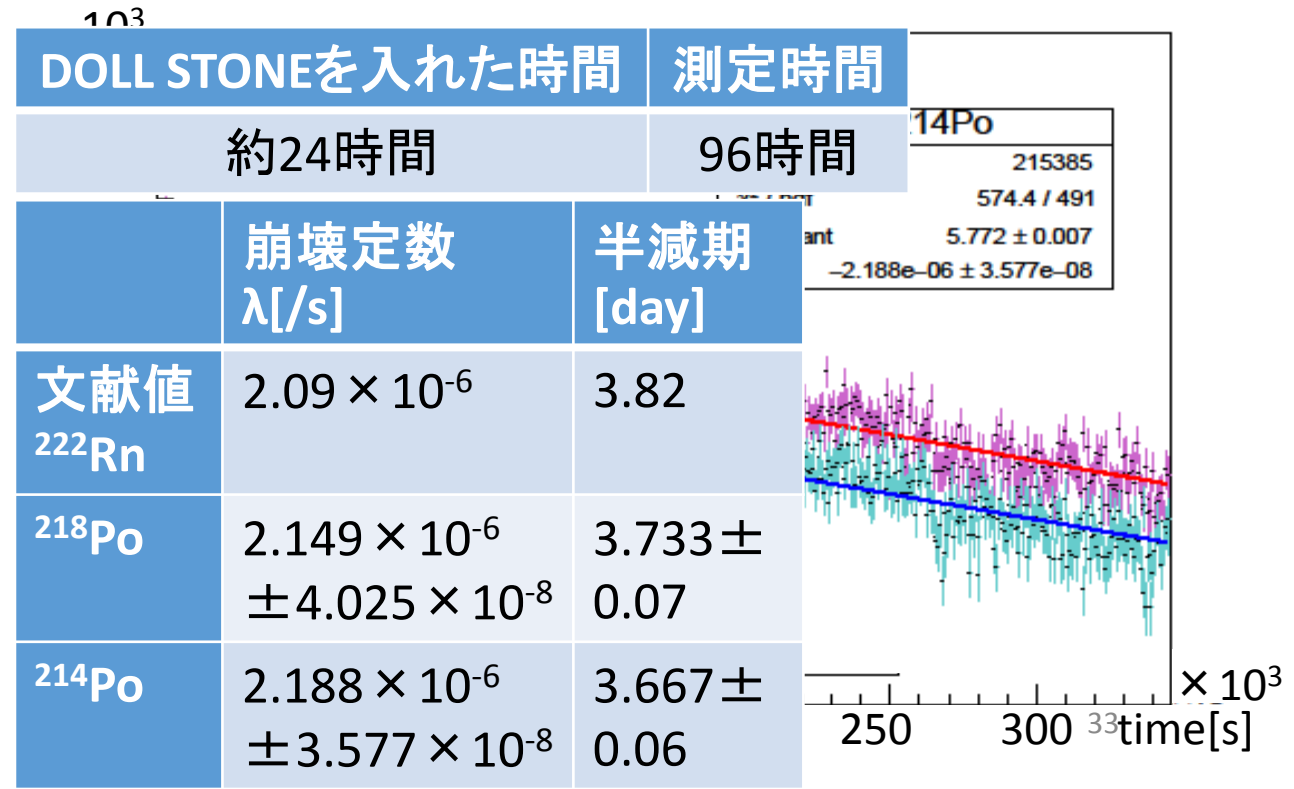
先行実験の小型ラドン検出器

Time-dN/dt



密封度を高めた小型ラドン検出器

Time-dN/dt



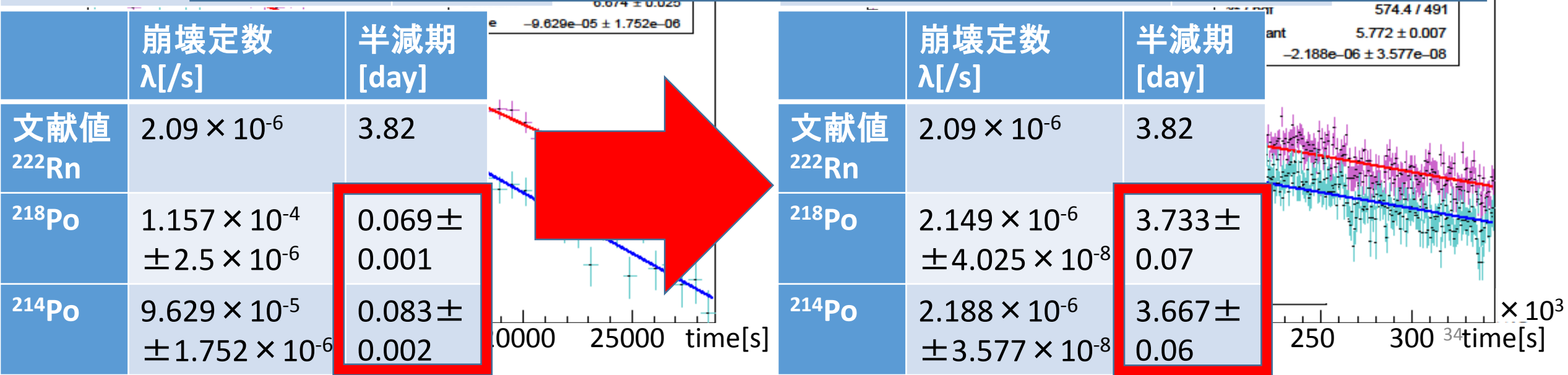
先行実験との比較

左は先行実験で作成した小型ラドン検出器での測定結果、右は体積がほぼ同じで密封度を高める工夫をした小型ラドン検出器での測定結果である。(1bin=10分)

先

測定結果から求めた半減期が Rnの半減期と同じオーダーとなった！！

DOLL STONE
約50

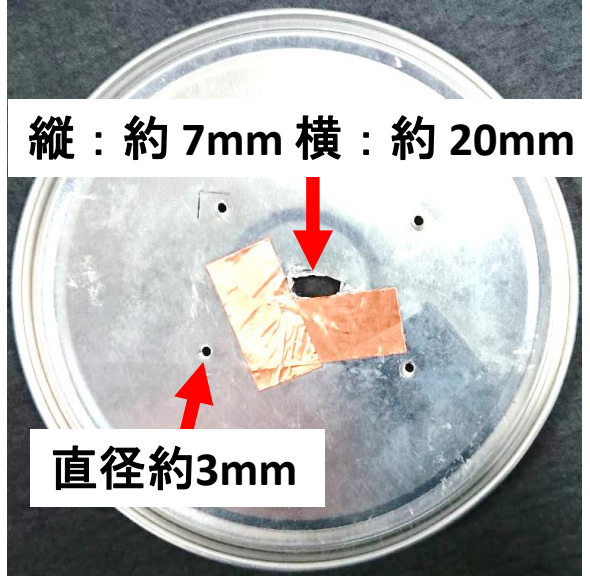


	崩壊定数 λ [s]	半減期 [day]
文献値 ^{222}Rn	2.09×10^{-6}	3.82
^{218}Po	1.157×10^{-4} $\pm 2.5 \times 10^{-6}$	0.069 ± 0.001
^{214}Po	9.629×10^{-5} $\pm 1.752 \times 10^{-6}$	0.083 ± 0.002

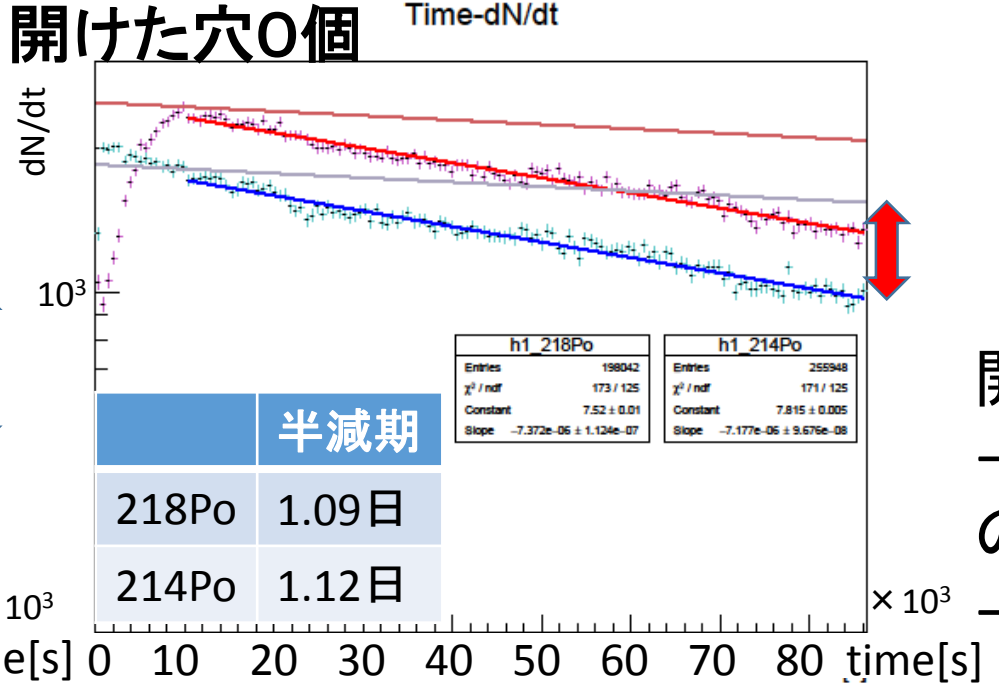
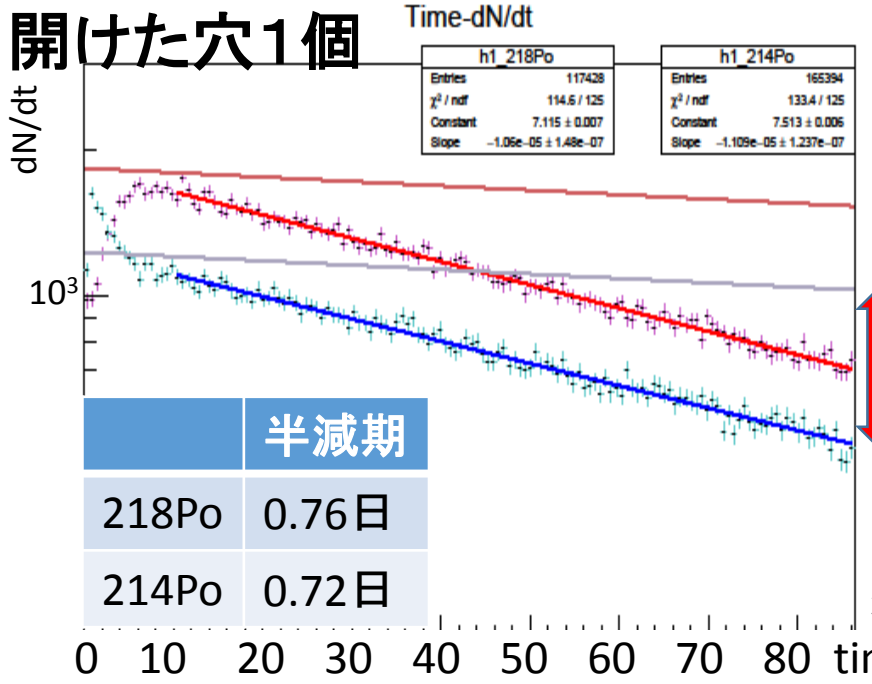
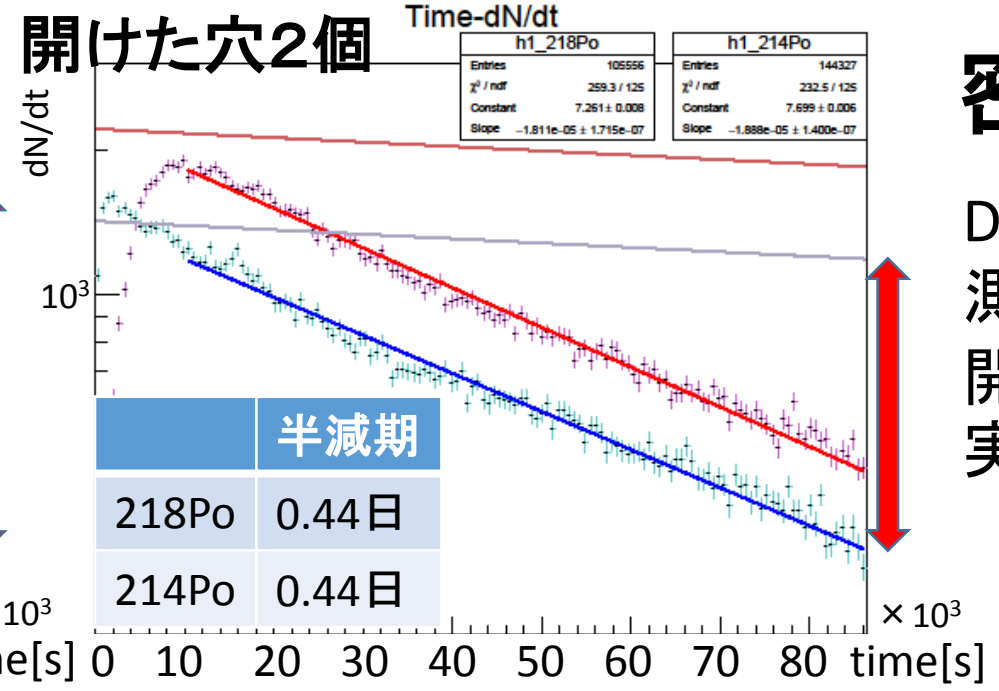
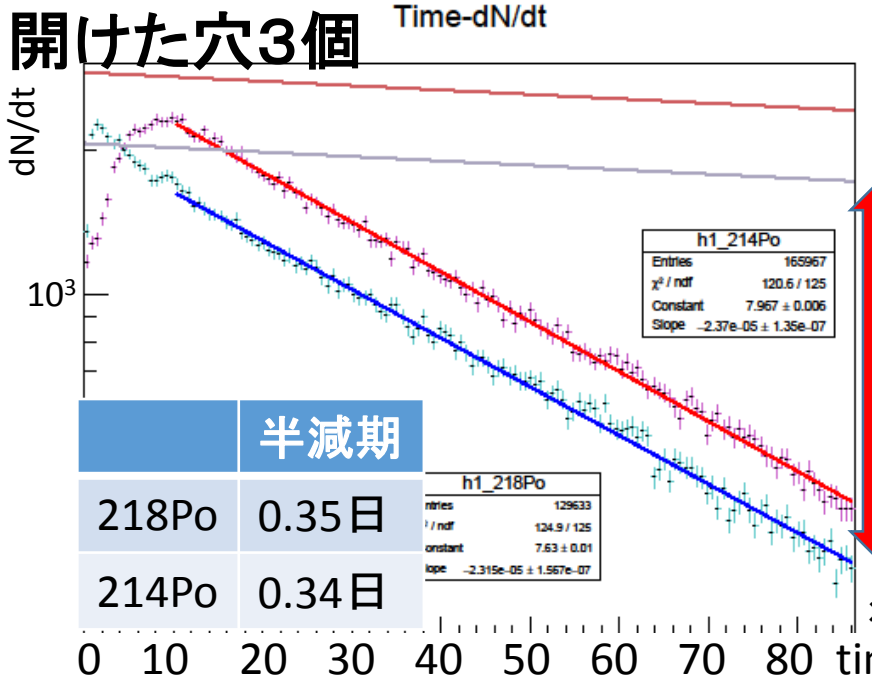
	崩壊定数 λ [s]	半減期 [day]
文献値 ^{222}Rn	2.09×10^{-6}	3.82
^{218}Po	2.149×10^{-6} $\pm 4.025 \times 10^{-8}$	3.733 ± 0.07
^{214}Po	2.188×10^{-6} $\pm 3.577 \times 10^{-8}$	3.667 ± 0.06

密封度の効果

DOLL STONE入れる時間・測定時間を同じにして開けた穴の数を考える実験を行った。



開けた穴が少なくなる
→fit線の傾きが文献値の傾きと差が小さくなる。
→**缶の密封度が重要**



まとめ

- ✓ 静電捕集法でPoが放出する α 線を検出することで間接的にラドンを検出
- ✓ 検出器を用いてラドンガスの測定、二つのピークを検出
- ✓ ^{241}Am 線源を用いてエネルギーと波高の関係を表す較正直線を導出→較正直線より二つのピーク= ^{218}Po と ^{214}Po
- ✓ ^{218}Po , ^{214}Po の単位時間当たりの崩壊数を調べて半減期を導出
- ✓ 密封度を高めることで検出器の性能がよくなる

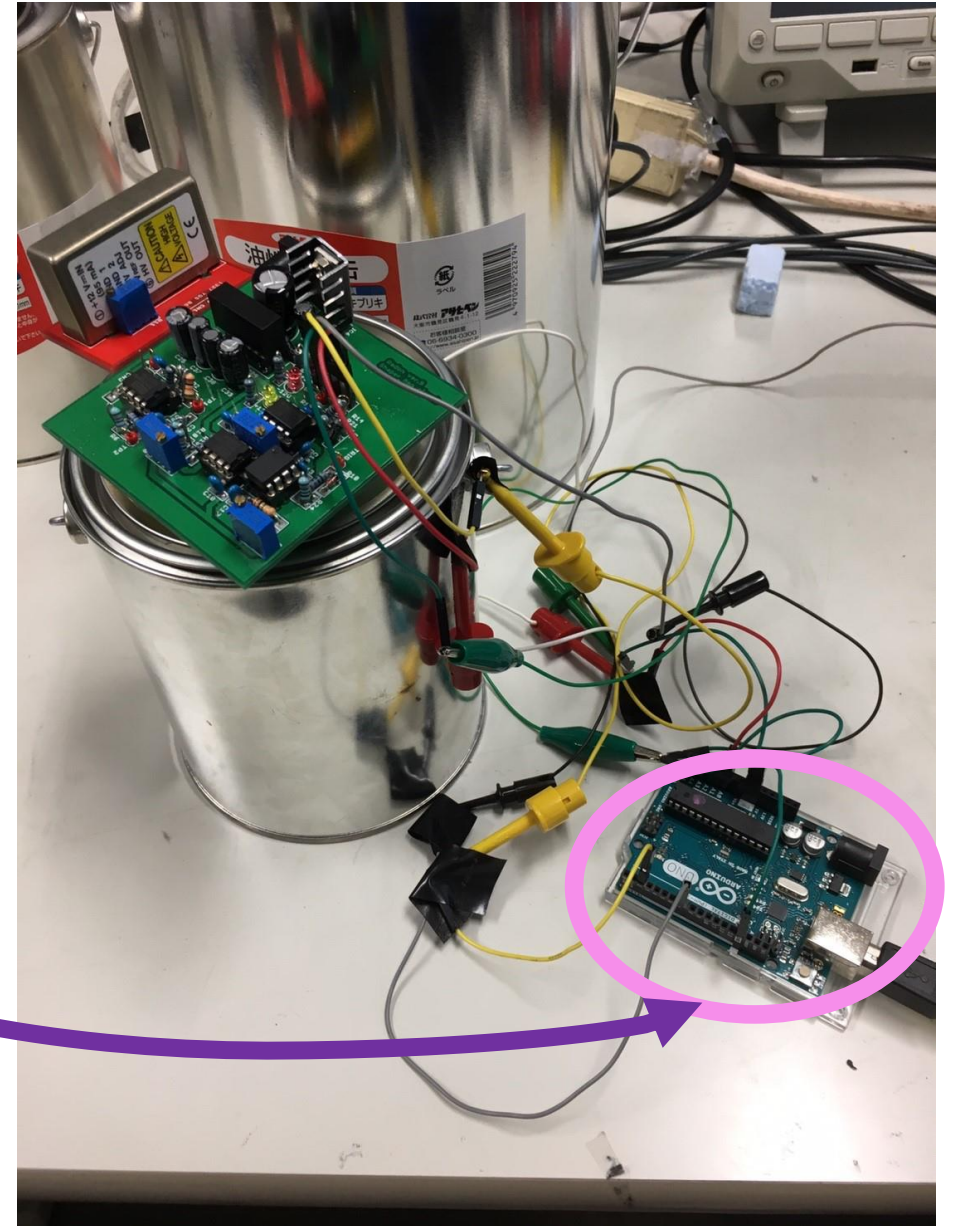
今後の展望

- ✓より性能の良い検出器にするため密封度を高める
- ✓検出器のさらなる小型化
- ✓環境変化による測定精度の変化の有無を調べるため温度・湿度を変えて測定

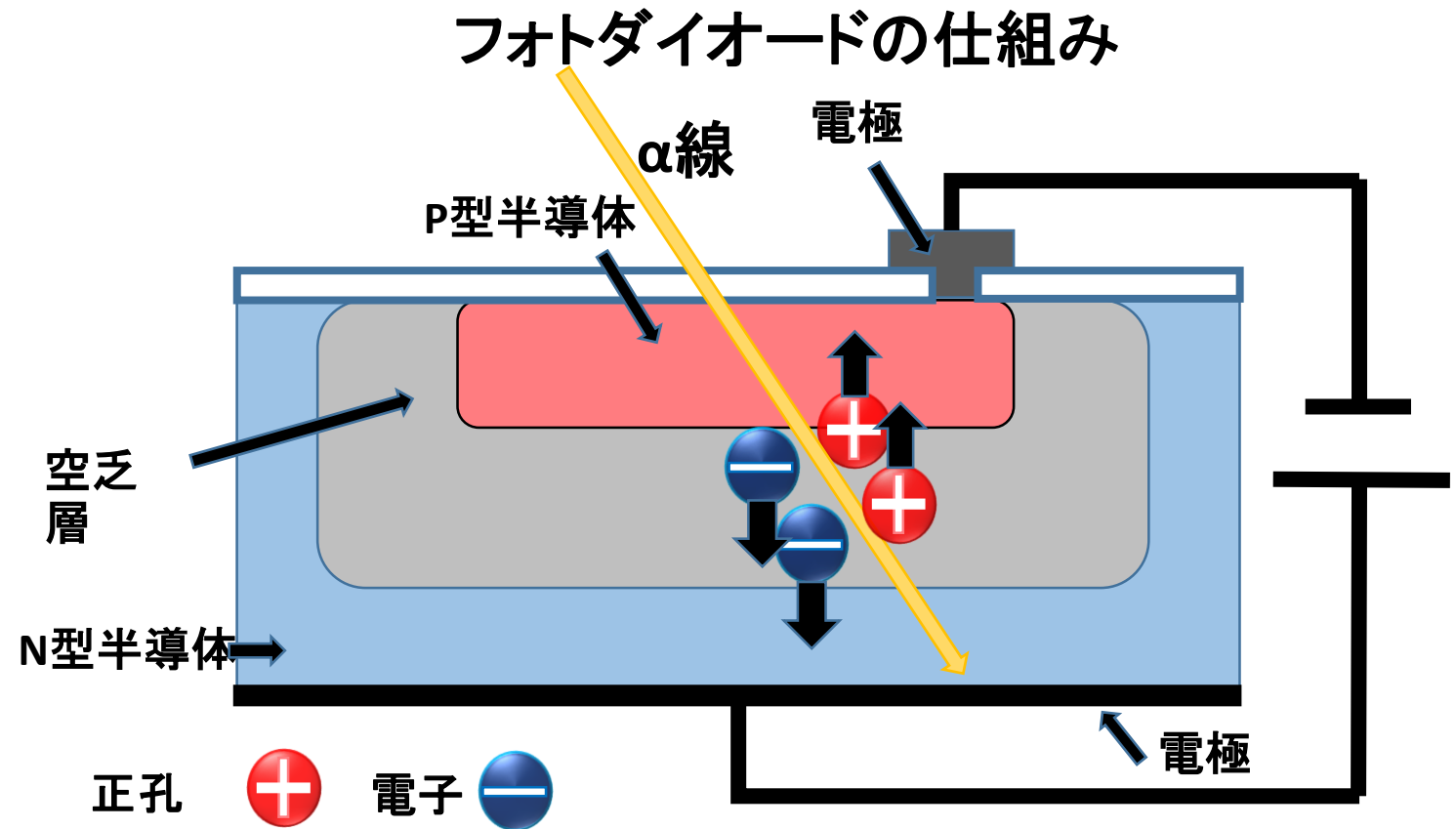
ご清聴ありがとうございました！

おまけ

ラドン検出の全体部



ラドン検出部



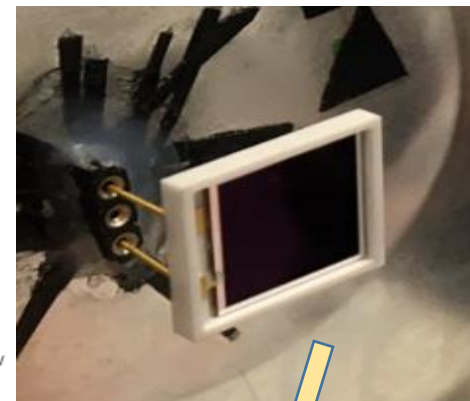
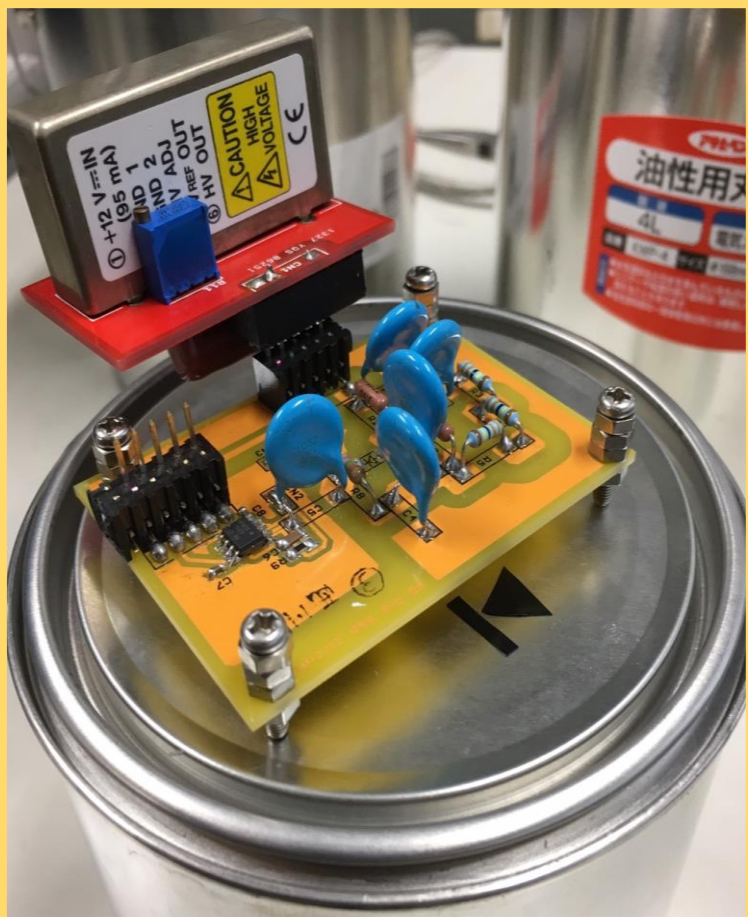
Siでは、一個の電子-正孔対を作るには約3.6(eV)のエネルギーが必要。

$$\alpha\text{線のエネルギー} \rightarrow 5.998 \times 10^6 \div 3.6(\text{eV}) = 1.7 \times 10^6(\text{個})$$

電子一個がもつ電荷の絶対値は $1.602 \times 10^{-19}(\text{C})$

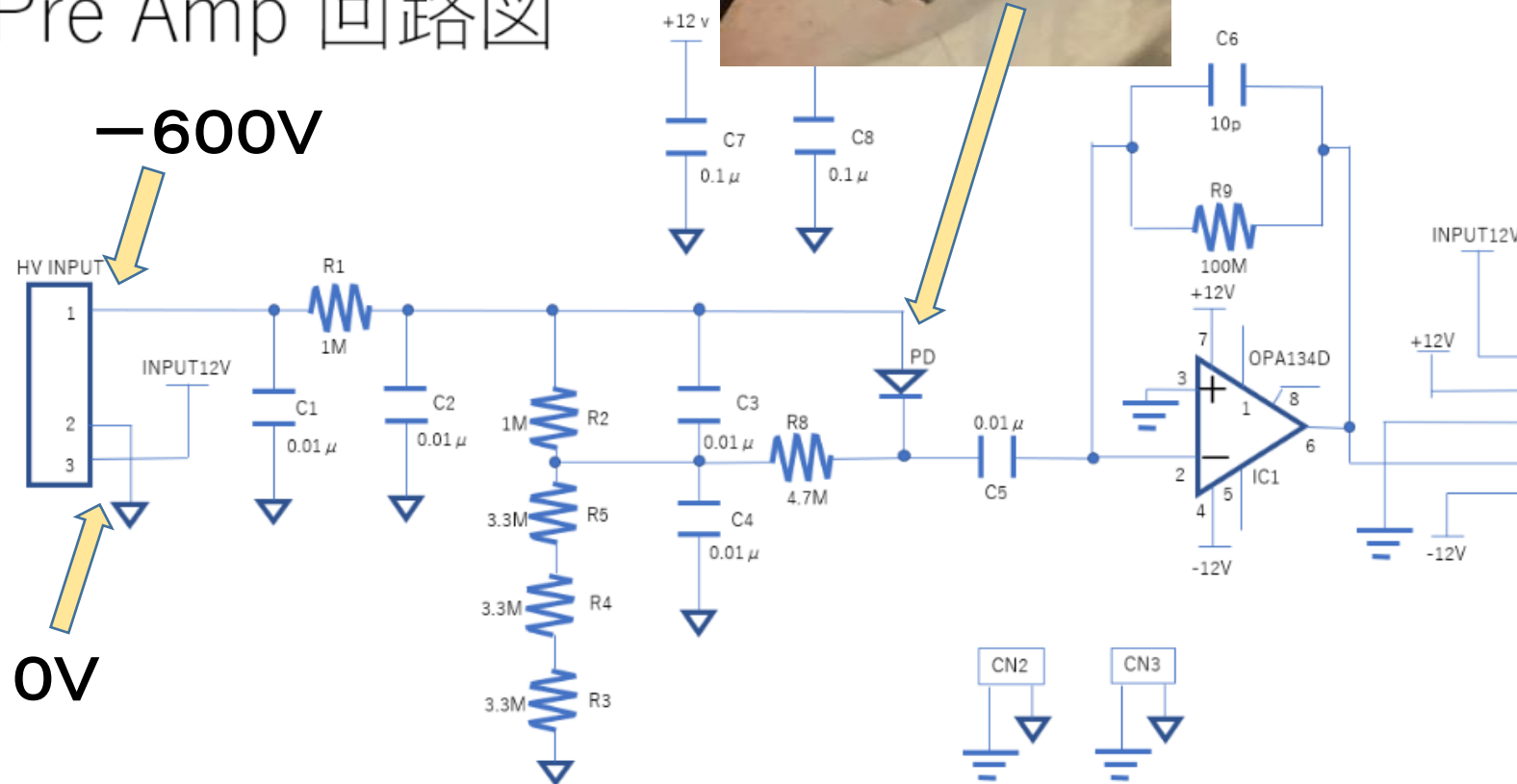
$$1.7 \times 10^6(\text{個}) \times 1.602 \times 10^{-19}(\text{C}) = 2.7 \times 10^{-13}(\text{C})$$

波高分析部～プリアンプ～



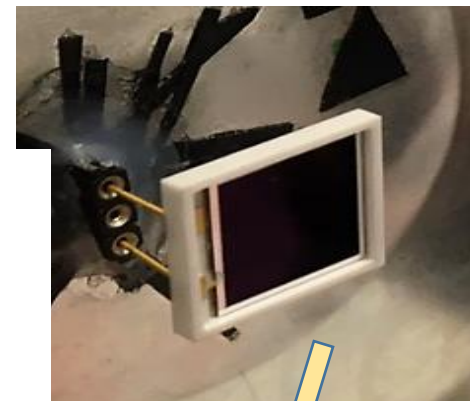
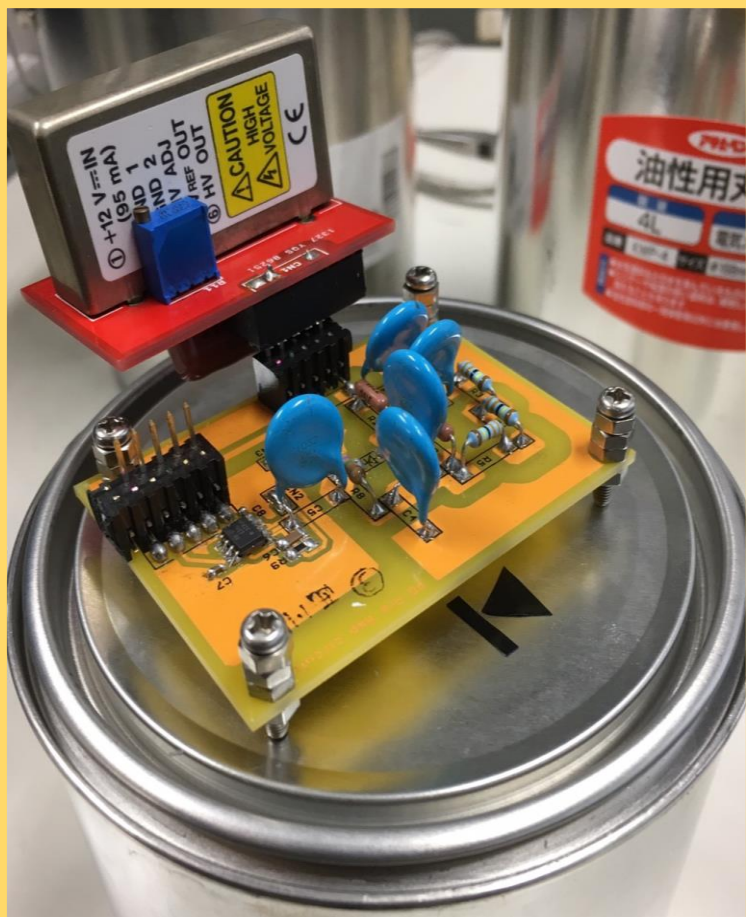
フォトダイ
オード(PD)

Pre Amp 回路図

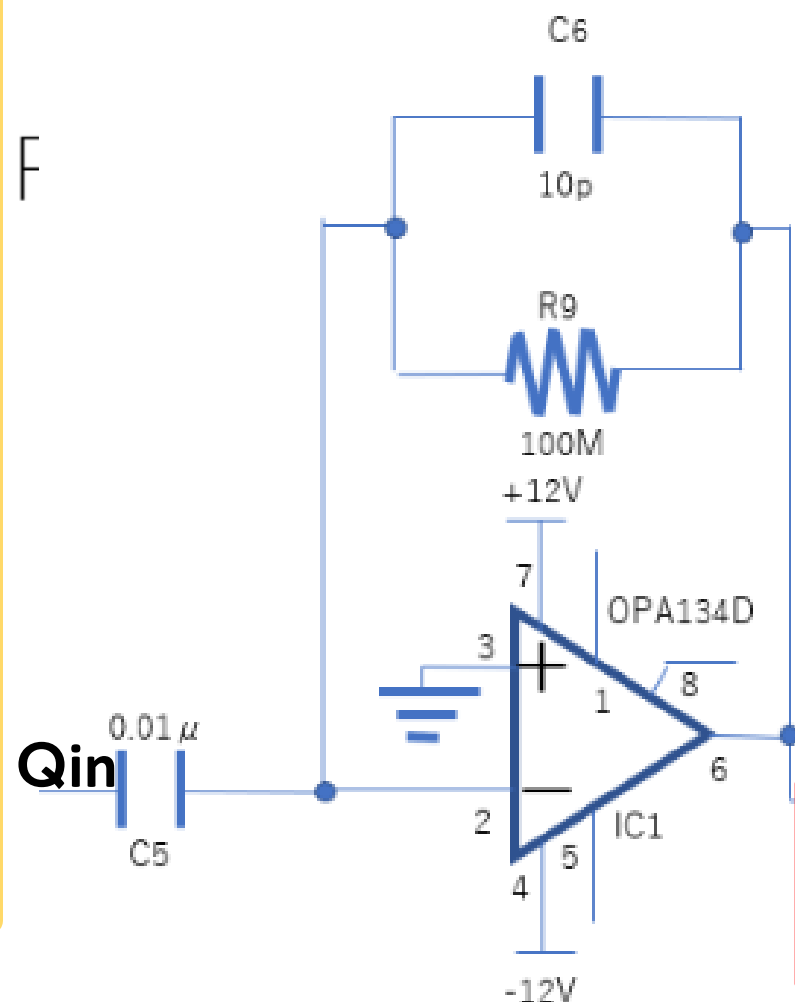


シミュレーター
アンプへの出力

波高分析部～プリアンプ～



フォトダイ
オード(PD)



$$Q = CV$$

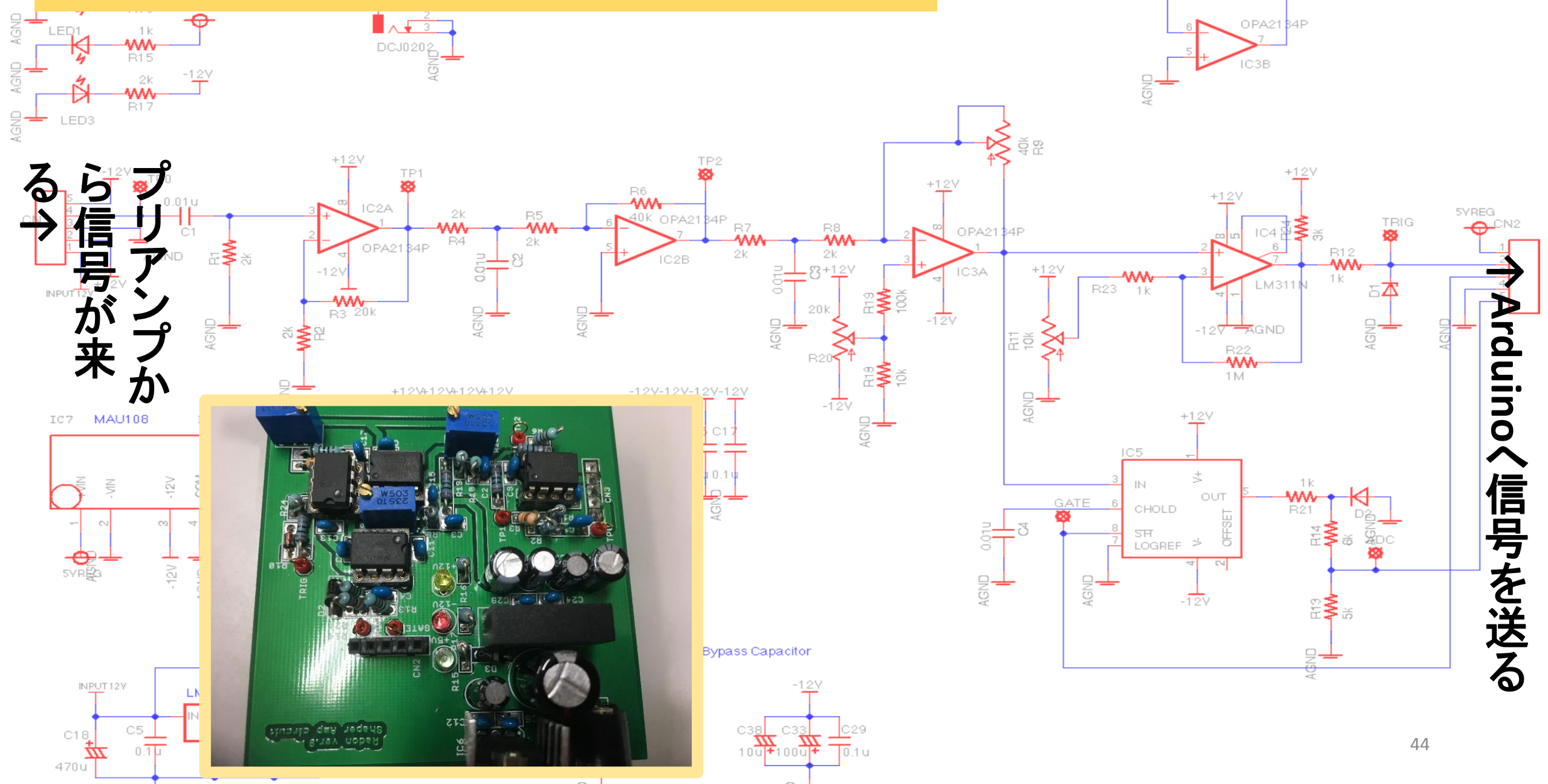
$$V_{out} = \frac{Q}{C} = 2.7 \times \frac{10^{-13} (C)}{10^{-11}} = 27mV$$

電圧が小さいので、
増幅をしないとイケない.....

波高分析部～シェイパーアンプ～

信号が来るからシェイパーアンプが

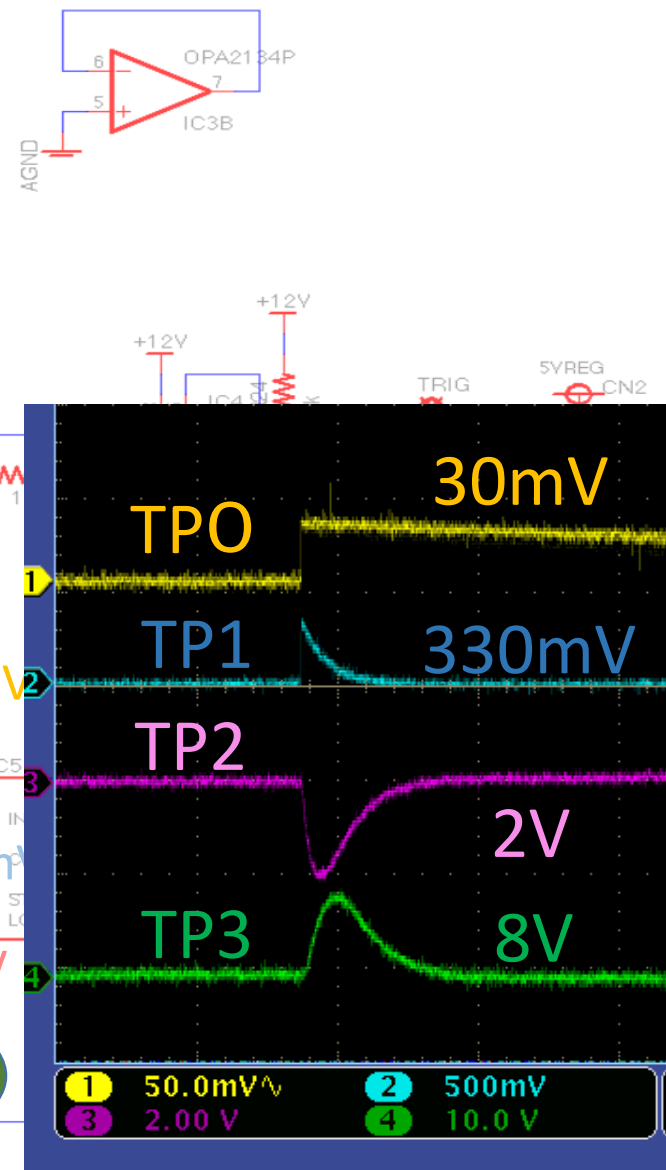
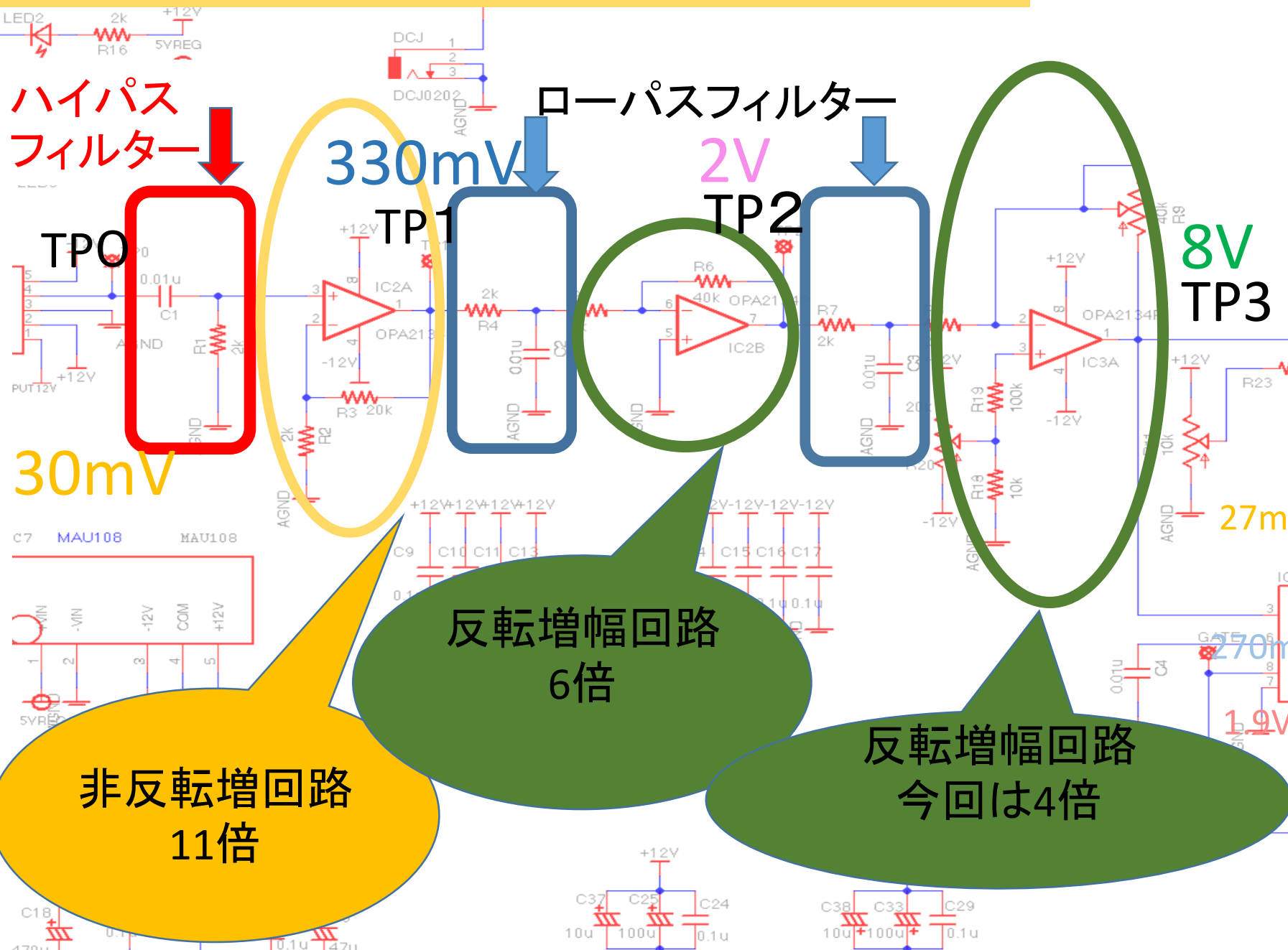
Arduinoへ信号を送る



Bypass Capacitor

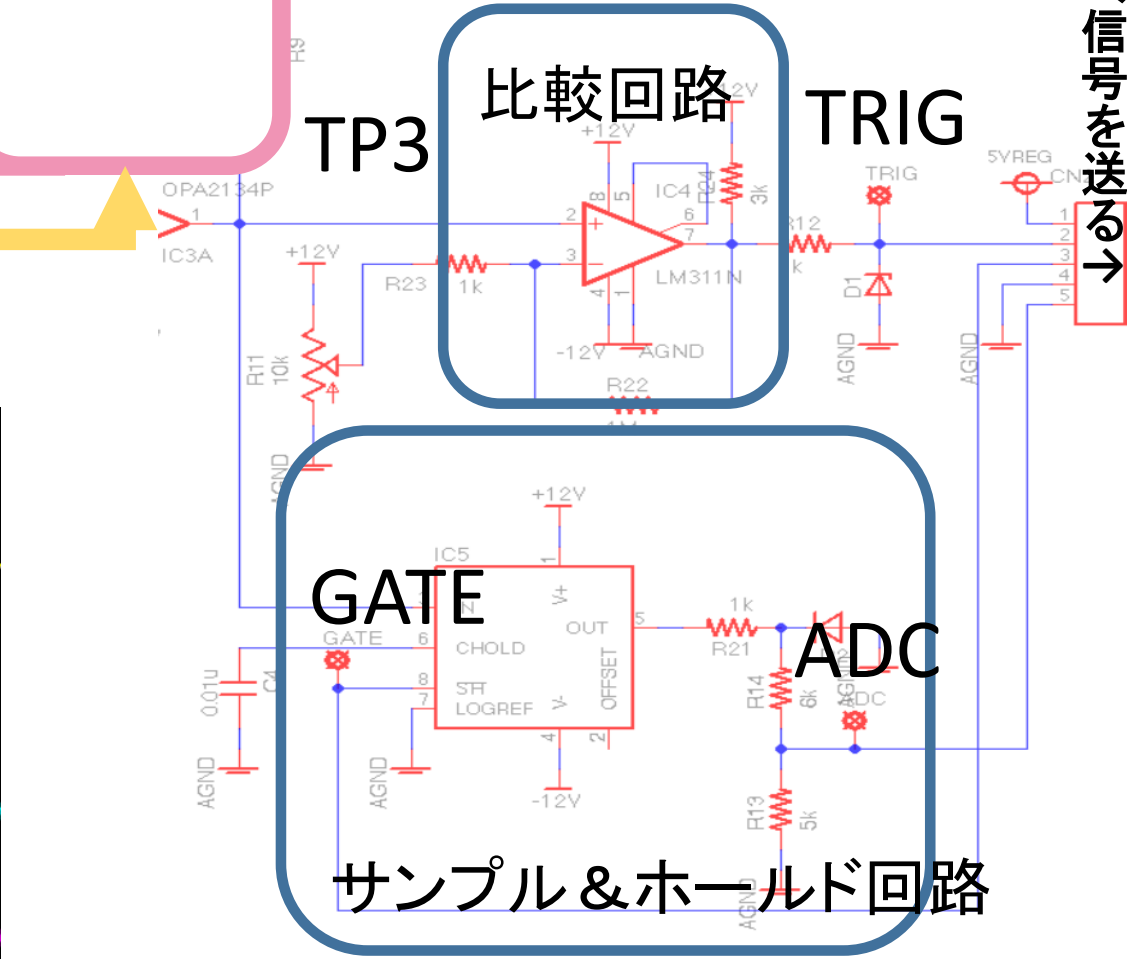
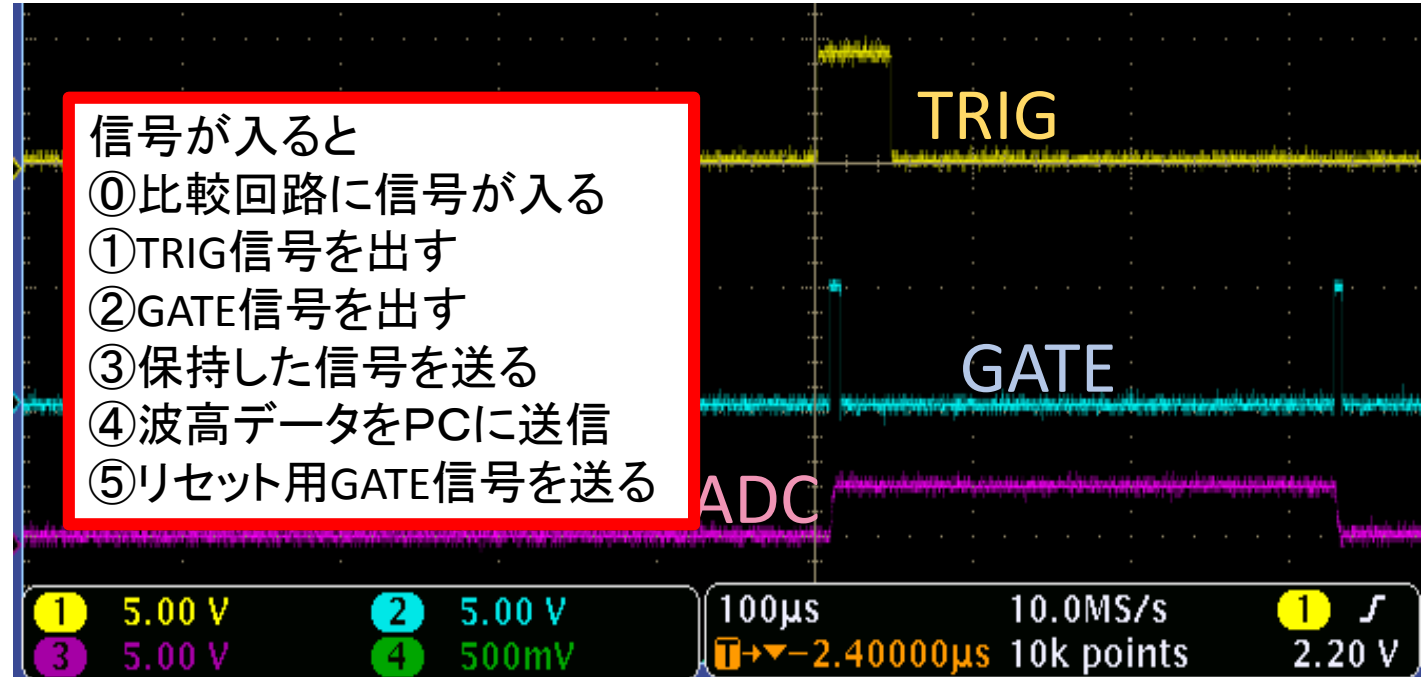
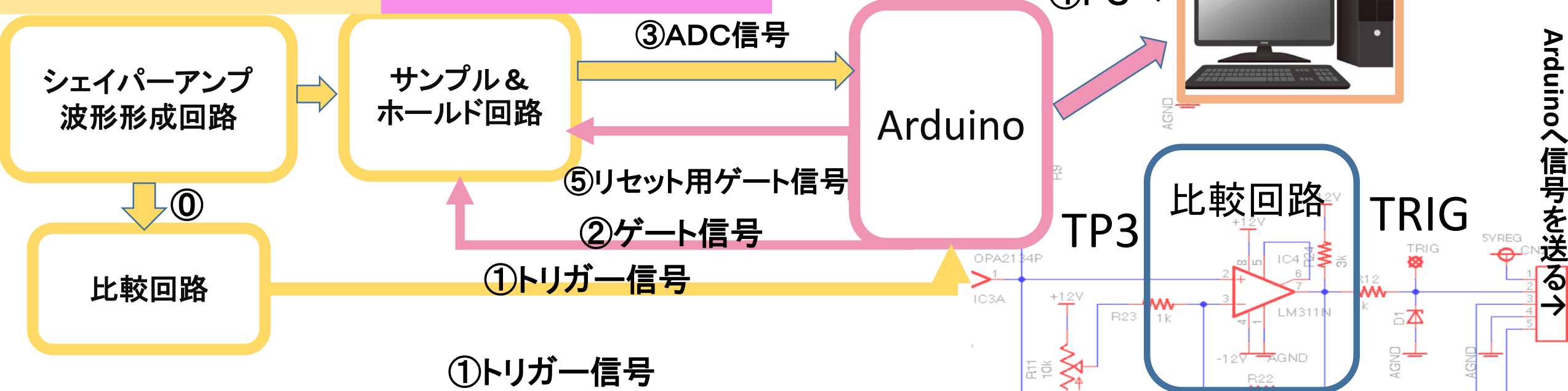
波高分析部～シェイパーアンプ～

→ プリアンプから信号が来る



サンプル&ホールド回路

波高分析部 + データ制御部



オンラインモニターの様子

Arduinoの基盤



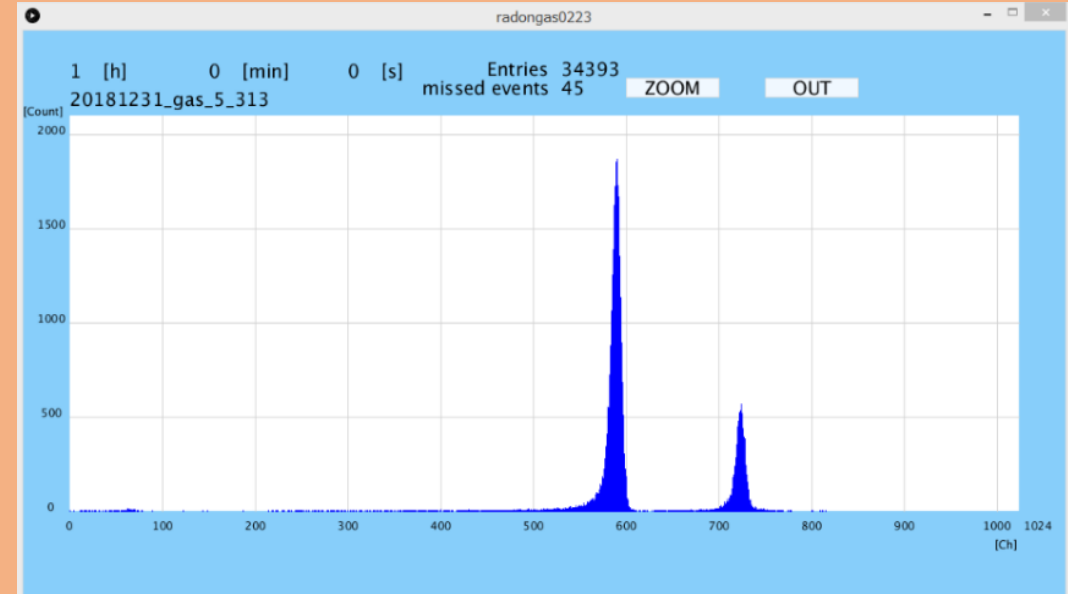
Processing

```
radongas0223 | Processing 3.4
import processing.serial.*; //Serial libraryを読み込む
//int totalTime = 1000; // (s)
//int totalTime = 1000000; // (s)
int startTime;
int currentTime;
int entry;
int missedEvent;
int raw_data;
int isHoldData;
int[] data = new int[1024];

String graph_t = "radongas0223"; //リアルタイムで更新されるグラフの表示タイトル
String filename = "radongas0223.txt"; //保存するファイルの名前

//GUI variables for data monitoring
int lens = 5; //拡大縮小倍率
float zoom_max = 125.0;
float zoom_min = 0.04;
float fZoom = 1.0;
float plotX1, plotX2;
float plotY1, plotY2;
float labelX, labelY;
```

データを送る

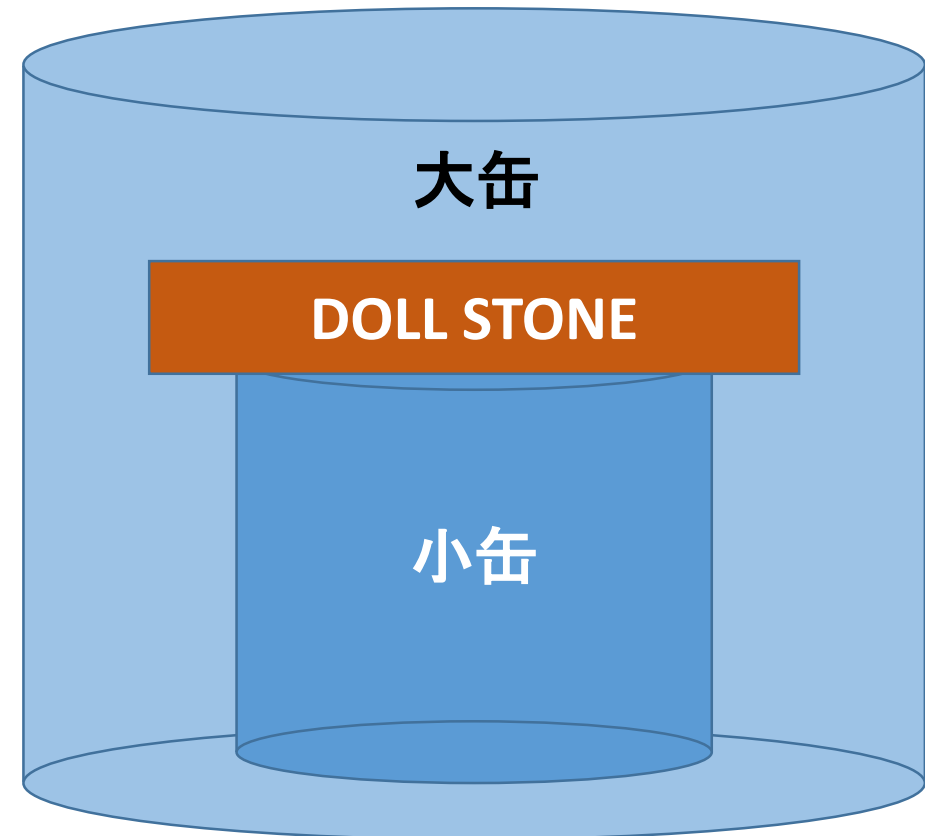
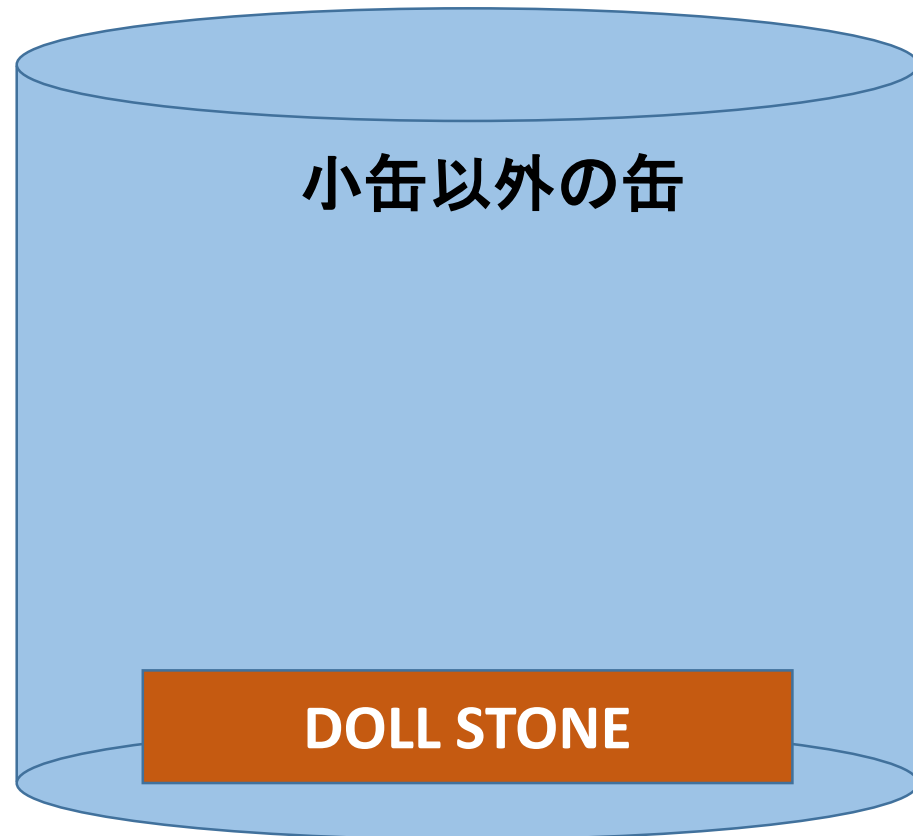


Txtファイル

TIME(s)	PulseHeight (ch)
1	33
1	46
1	76
1	92
1	90
1	67
1	50
1	40
1	28
1	38
1	38
1	55
1	55
1	36
1	41

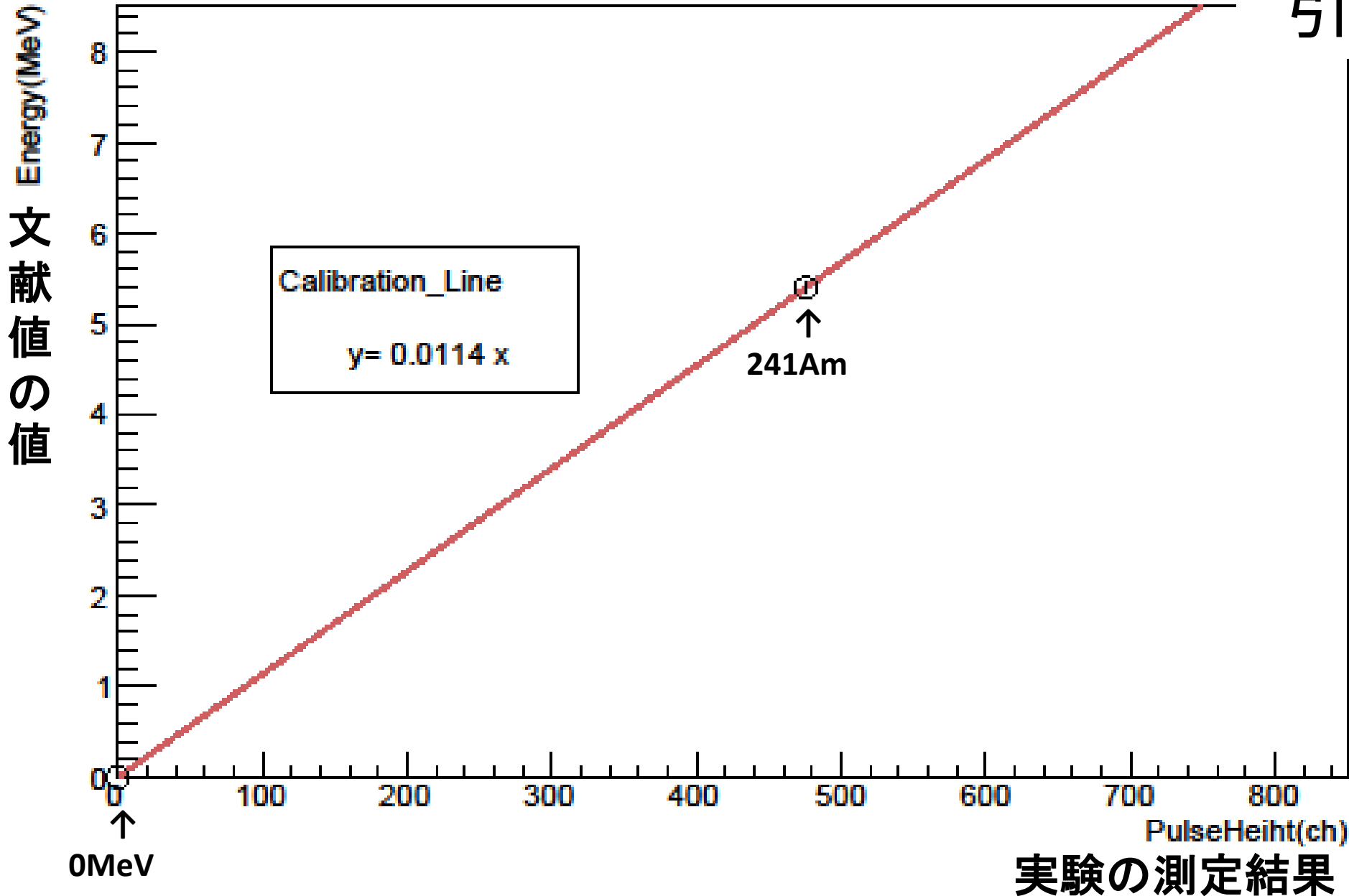
小缶でのラドンガス 封入方法

小缶には直接DOLL STONEが入らなかったため下の右図のようなセットアップでラドンガスを封入した。



Energy-PulseHeight

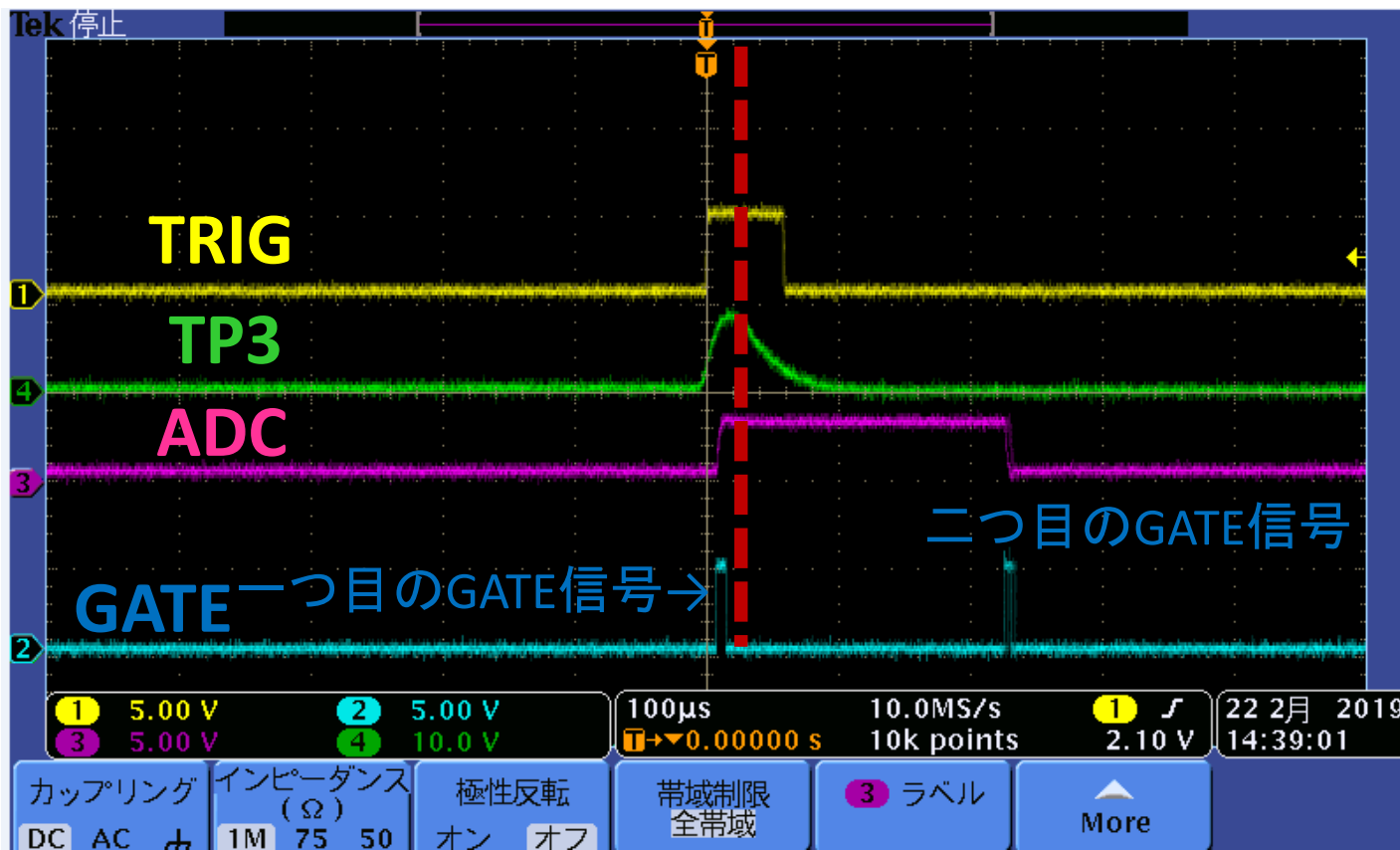
Am線源を用いて 引いた較正直線



0MeVのchを0chとした時のy座標を α 崩壊したときの α 線の文献値のエネルギー、x座標を実験で求めたchとして0MeV,241Amの点をプロットし、その2点を用いて求めた較正直線を引いた図である。

この較正直線によって、エネルギーとch数の関係を大まかに知ることができる。

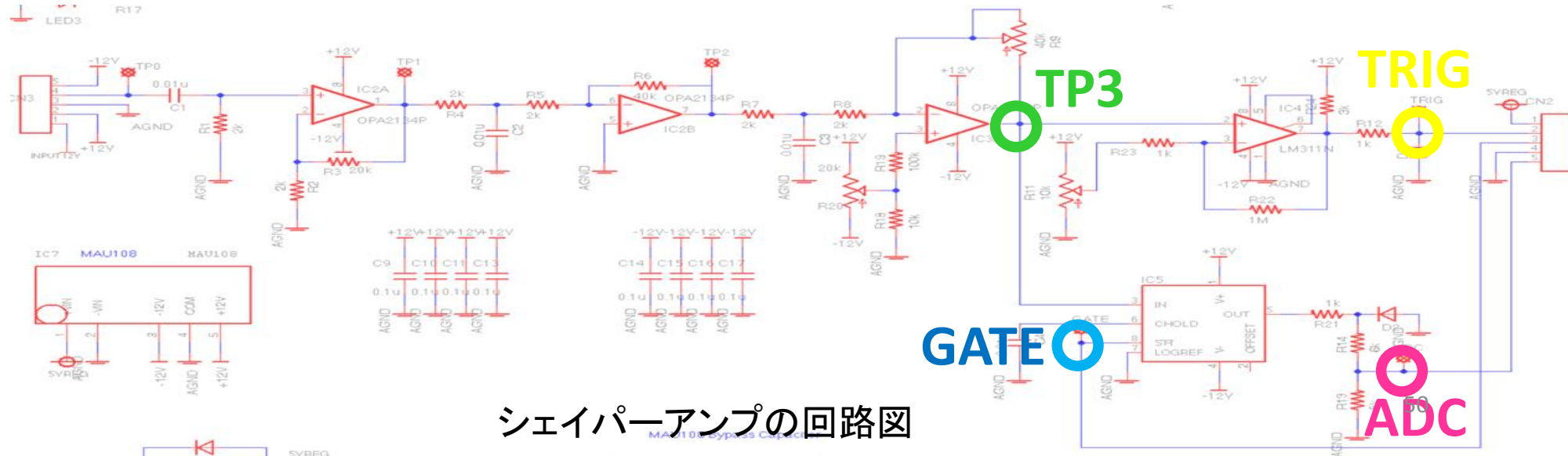
オシロスコープの波形



ArduinoはTRIG (TRIGGER) 信号を受け取るとサンプル&ホールド回路へGATE信号を送る。GATE信号を受け取ったサンプル&ホールド回路は、一つ目のGATE信号の立ち下がりの時のTP3の値を読み込む。その値を二つ目のGATE信号がくるまで保持している信号がADC信号である。

一つ目のGATE信号の幅を変えることによって読み込む値を変えることができる。

↑オシロスコープの写真



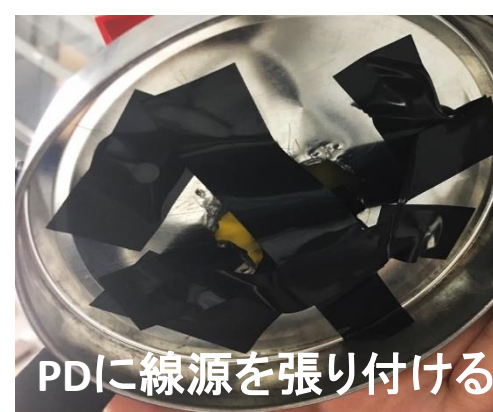
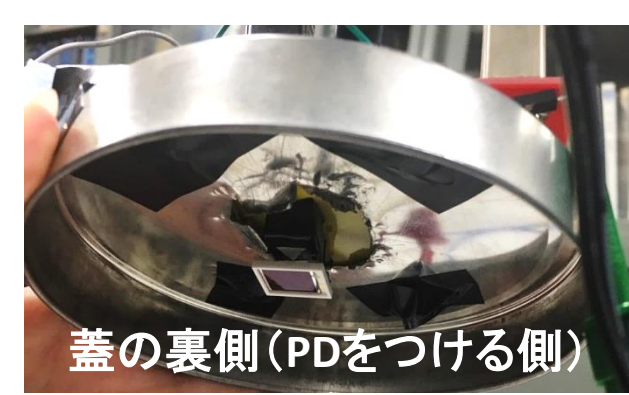
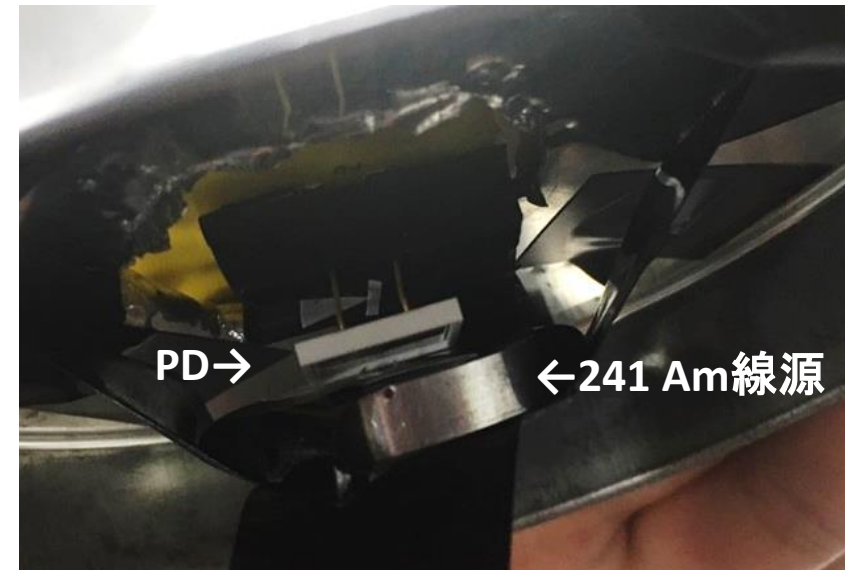
シェイパーアンプの回路図

Am線源の測定

chとエネルギーの関係を調べるために、

線源を用いて0点時と線源のピーク時のchを測定した。

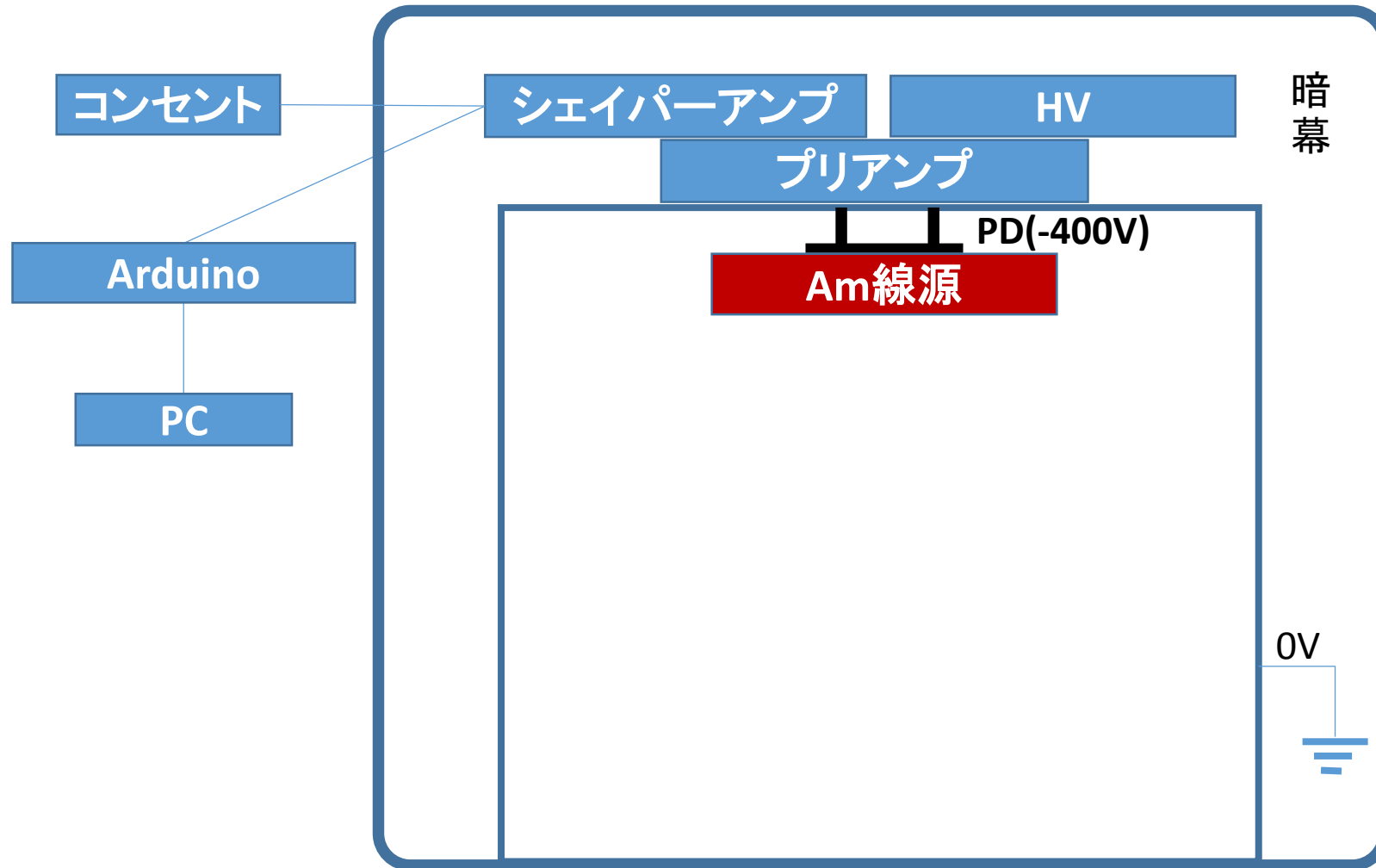
今回使用した ^{241}Am 線源の α 線のエネルギーは5.4MeVである。



蓋の裏側(PDをつける側)

241 Am線源

PDに線源を張り付ける



コンセント

シェイパーアンプ

HV

暗幕

プリアンプ

PD(-400V)

Am線源

Arduino

PC

0V

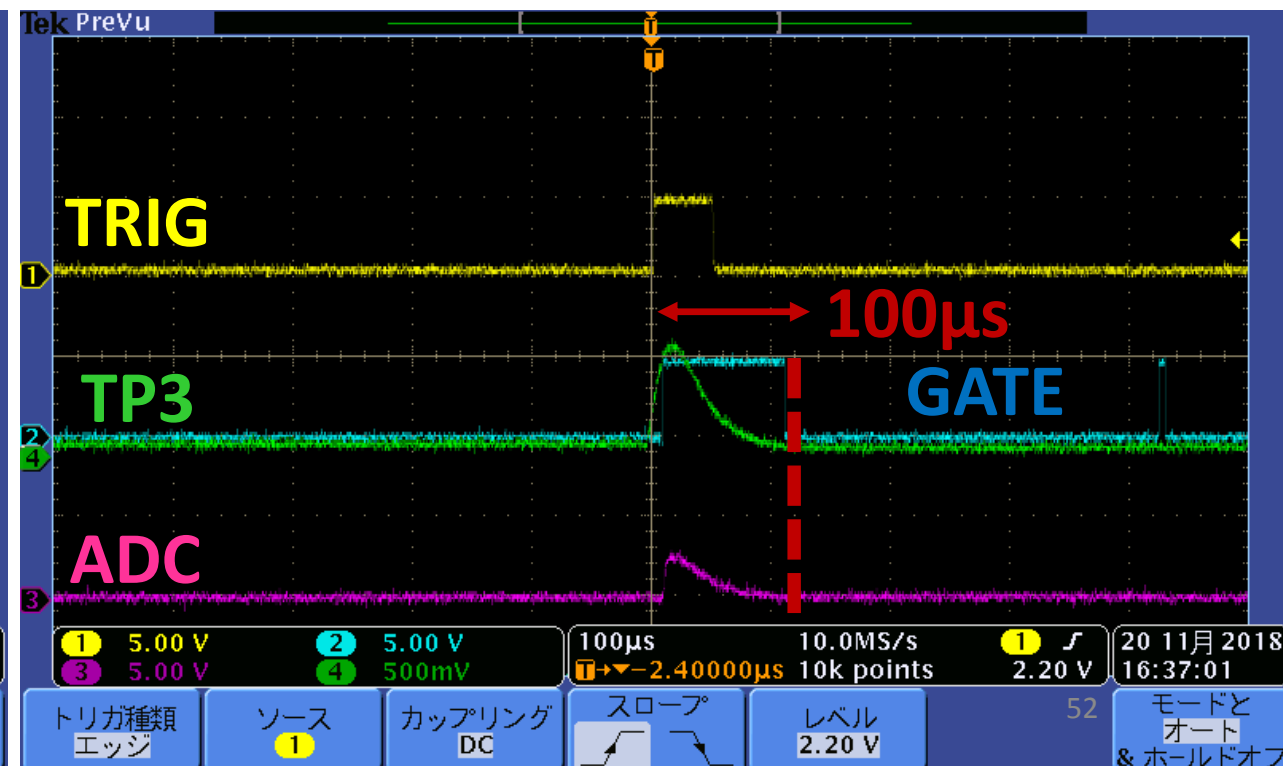
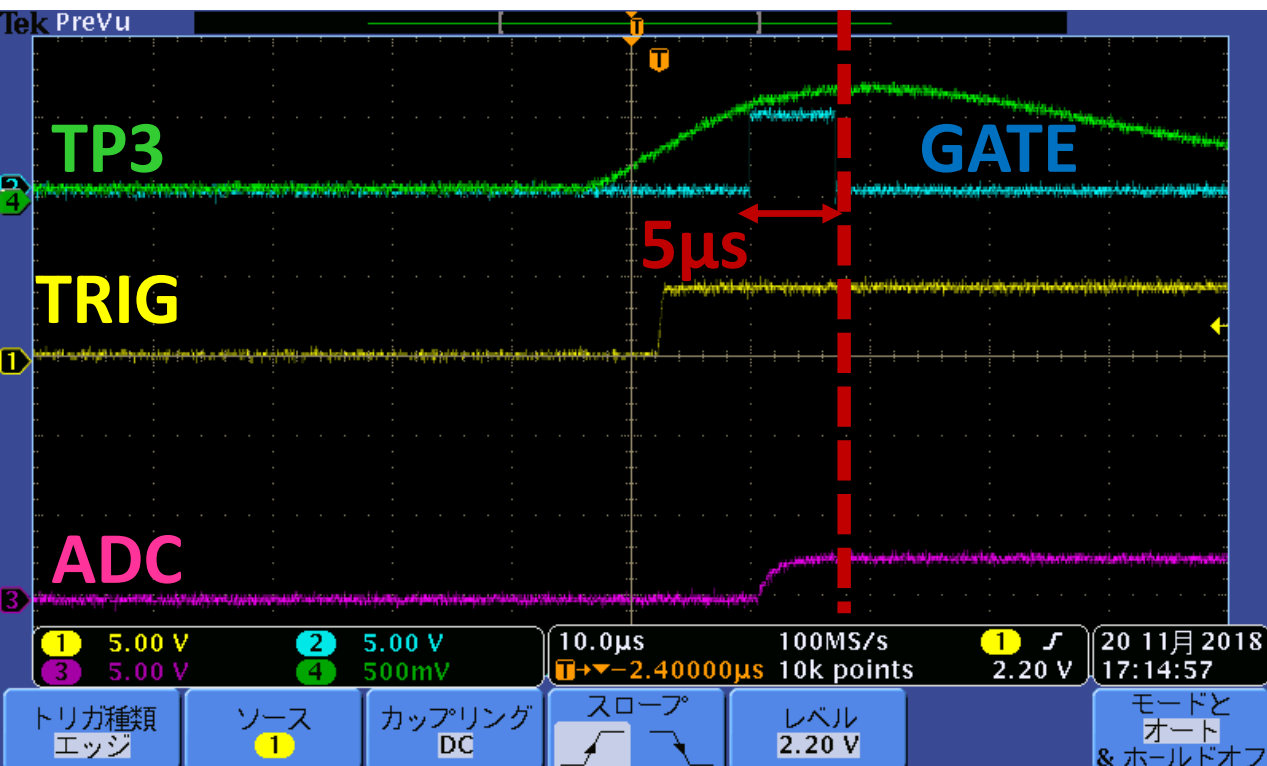
Am線源測定時の オシロスコープの波形

ラドン検出器では一つ目のGATE信号の立ち上がり時に波形整形された後の信号(TP3)の値を読み込んでいる。
よって、

一つ目のGATE幅を5 μ s に設定すると、TP3のピーク時の値を読み込み、
一つ目のGATE幅を100 μ s に設定すると、0点時の値を読み込むことができる。
Am線源を測定しているので、左ではAmの α 線のエネルギーを測定しており、
右では0MeVの時のADCの値を測定している。

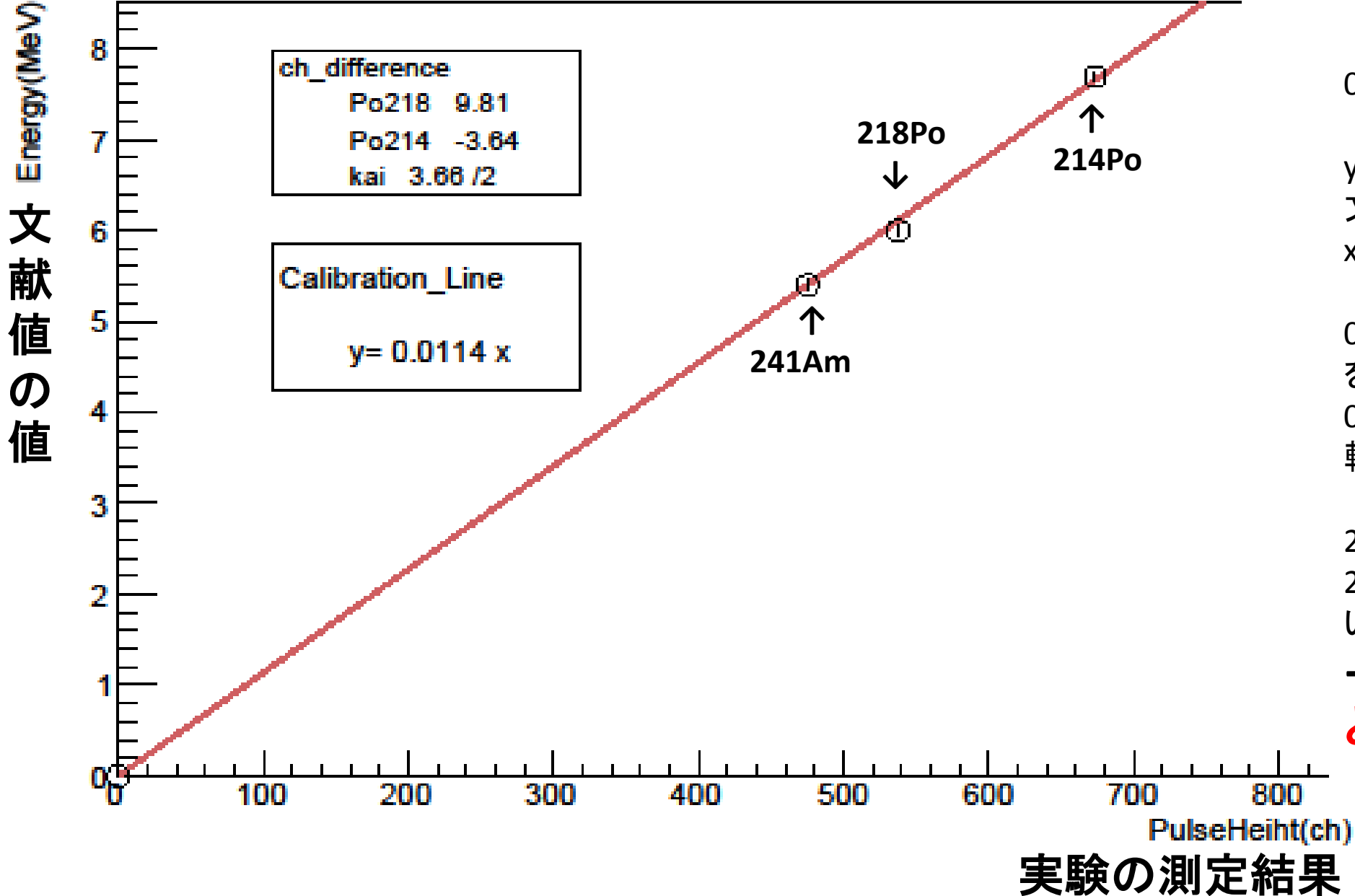
一つ目のGATE幅 5 μ s

一つ目のGATE幅 100 μ s



Energy-PulseHeight

Am較正直線との比較



0MeVのchを0chとした時の、
y座標を α 崩壊したときの α 線の
文献値のエネルギー、
x座標を実験で求めたchとして

0MeV, 241Am, 218Po, 214Poの点
をプロットし、
0MeVと241Amを用いて求めた
較正直線を引いた図である。

218Poは直線よりも右に、
214Poは直線よりも左にずれて
いる。

→218Poと214Poで直線
との関係が違う???

Am較正直線との比較

0MeVのchを0chとした時の、

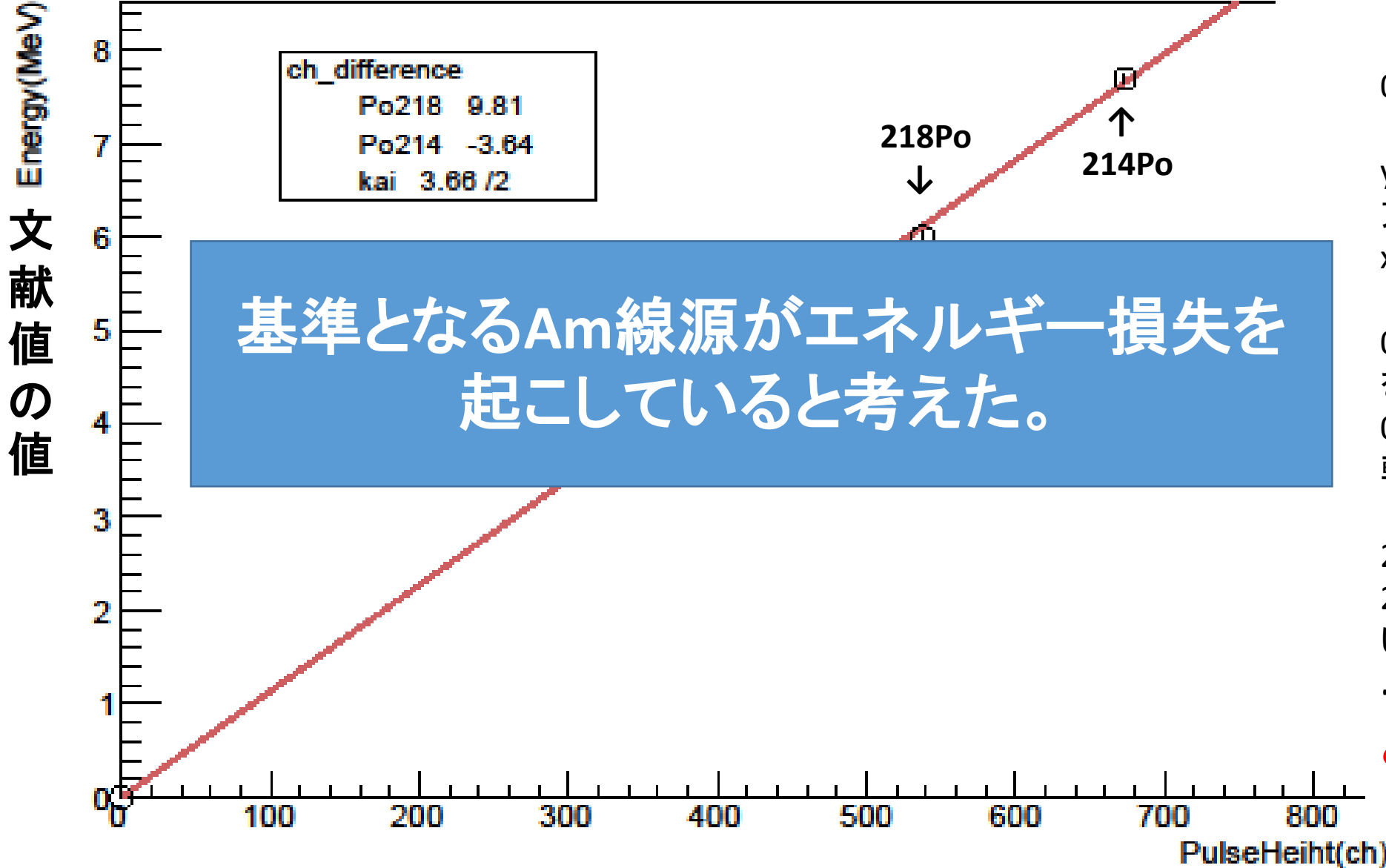
y座標を α 崩壊したときの α 線の文献値のエネルギー、x座標を実験で求めたchとして

0MeV, 241Am, 218Po, 214Poの点をプロットし、0MeVと241Amを用いて求めた較正直線を引いた図である。

218Poは直線よりも右に、214Poは直線よりも左にずれている。

→218Poと214Poで直線との関係が違う???

Energy-PulseHeight



Am線源のエネルギー損失

今回の実験では ^{241}Am 密封線源を用いた。

放射線及び放射能測定器の校正目的で使用される密封線源の窓には厚さ $1\sim 2\mu\text{m}$ の金、もしくはパラジウム、もしくは金パラジウム合金が使用される。

ベーテブロッホの式

$$-\frac{dE}{dx} = 4\pi N_a r_e^2 m_e c^2 \rho \frac{Z}{A} \frac{z^2}{\beta^2} \left[\ln \left(\frac{2m_e \gamma^2 v^2 W_{max}}{I^2} \right) - 2\beta^2 \right] [\text{MeV/cm}]$$

より、 α 線がAuを通過したときに単位長さ当たりで失うエネルギーは $8638.02 [\text{MeV/cm}]$

この部分に密封線源の窓として薄い膜が貼られている。



よって、密封線源の窓が厚さ $1\mu\text{m}$ の時

$$8638.02 \left[\frac{\text{MeV}}{\text{cm}} \right] \times 1 \times 10^{-4} [\text{cm}] = 0.0863802 [\text{MeV}]$$

のエネルギー損失を起こしている。

今回使用したAm線源が何で密封されているのか確認できなかったが、校正目的で使用される密封線源と同程度のエネルギー損失を起こしていると仮定し、Amの α 線のエネルギーを 5.3MeV として較正直線を引いた。

↑使用した ^{241}Am 線源

Am線源のエネルギー損失(おまけ)

ベーテブロッホの式

$$-\frac{dE}{dx} = 4\pi N_a r_e^2 m_e c^2 \rho \frac{Z}{A} \frac{z^2}{\beta^2} \left[\ln \left(\frac{2m_e \gamma^2 v^2 W_{max}}{I^2} \right) - 2\beta^2 \right] [\text{MeV/cm}]$$

$$4\pi N_a r_e^2 m_e c^2 = 0.3070 [\text{MeVcm}^2/\text{g}]$$

$$W_{max} \sim 2m_e c^2 \eta^2 \quad (\eta = \beta\gamma)$$

$$\frac{I}{Z} = 9.76 + 58.8Z^{-1.19} [\text{eV}] \quad (Z \geq 13)$$

Auの場合、 $\rho = 19.32 [\text{g/cm}^3]$, $Z = 79$, $A = 196.97$

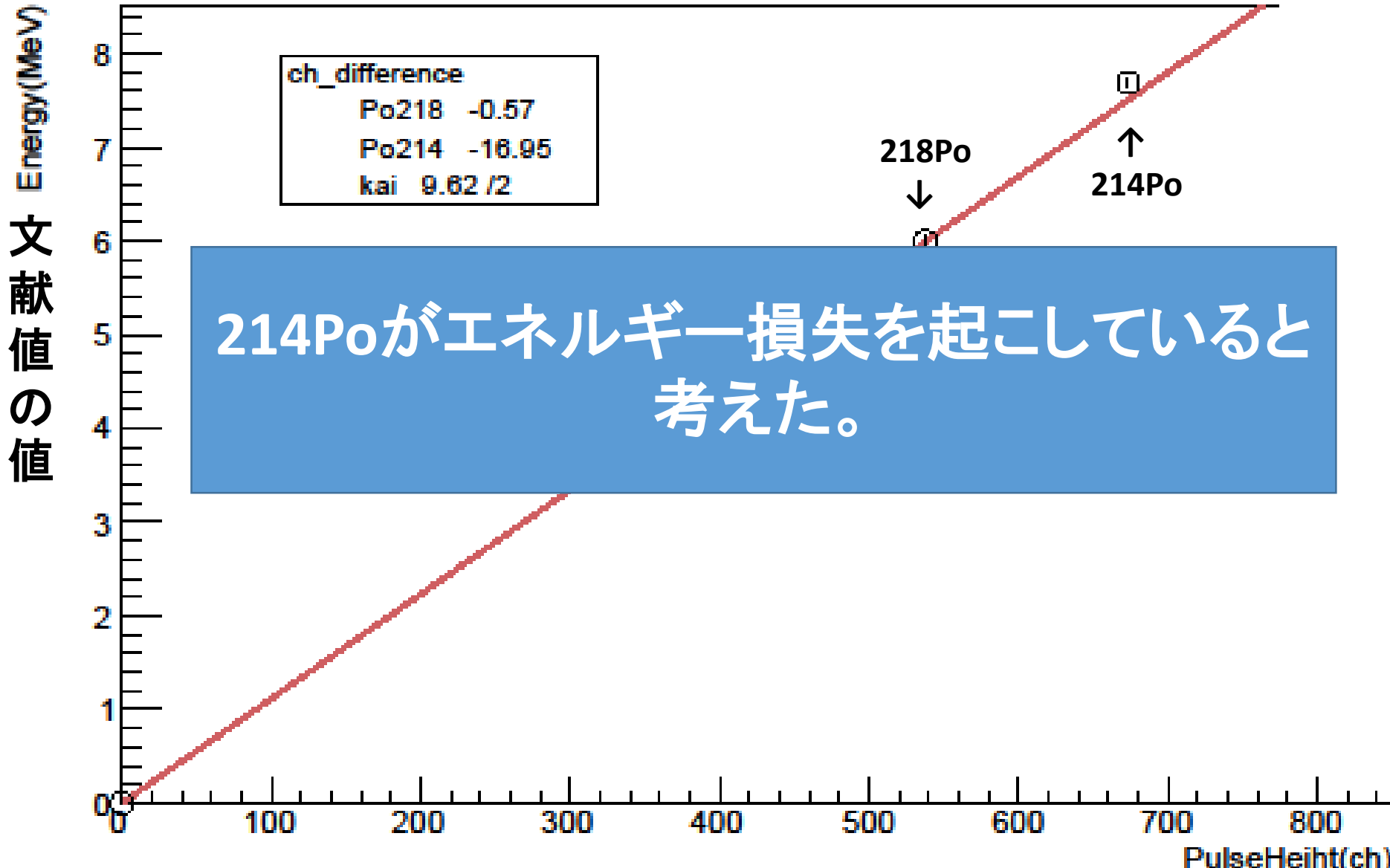
A線の速度は核種にもよるが、 $v = (1.5 \sim 2.0) \times 10^7 [\text{m/s}]$

z は α 線の電荷、 β は v/c 、 γ は $1/\sqrt{1-\beta^2}$

より計算すると、 α 線がAuを通過したときに単位長さ当たりで失うエネルギーは $8638.02[\text{MeV/cm}]$

Energy-PulseHeight

Am校正直線との比較



文献値の値

実験の測定結果

0MeVのchを0chとした時の、y座標を α 崩壊したときの α 線の文献値のエネルギー、x座標を実験で求めたchとして

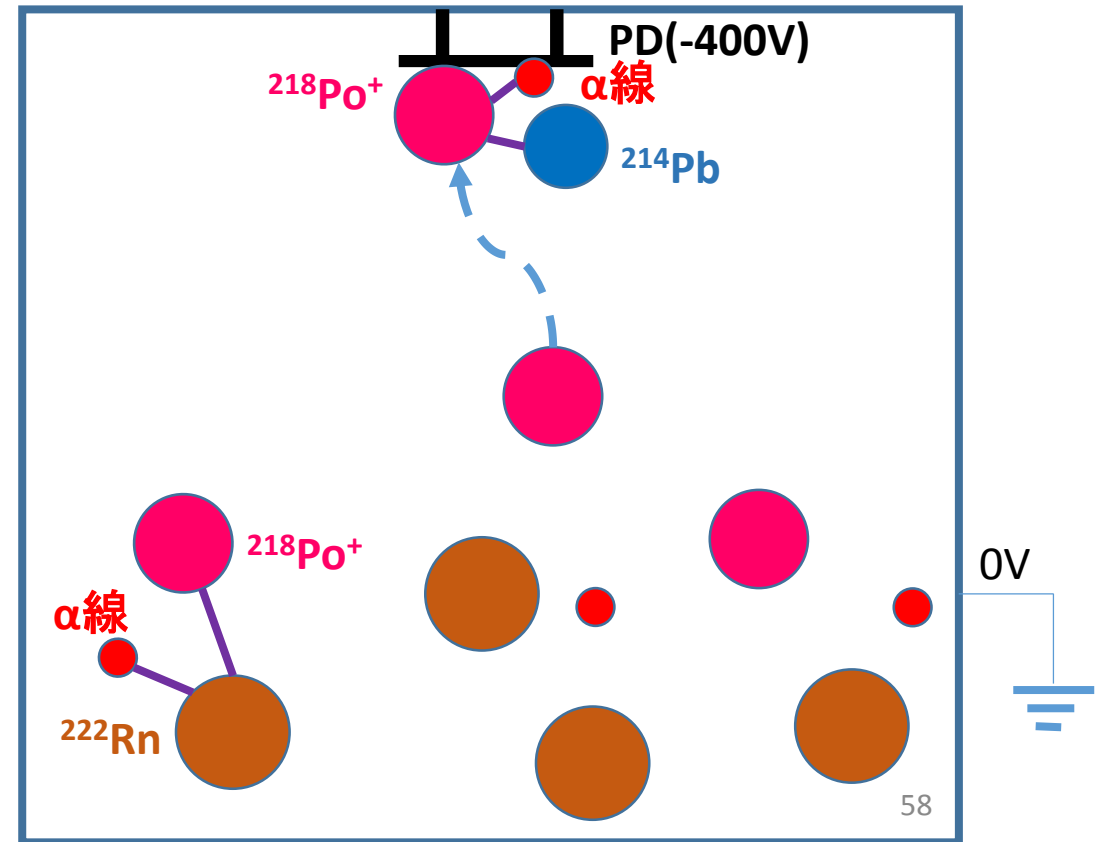
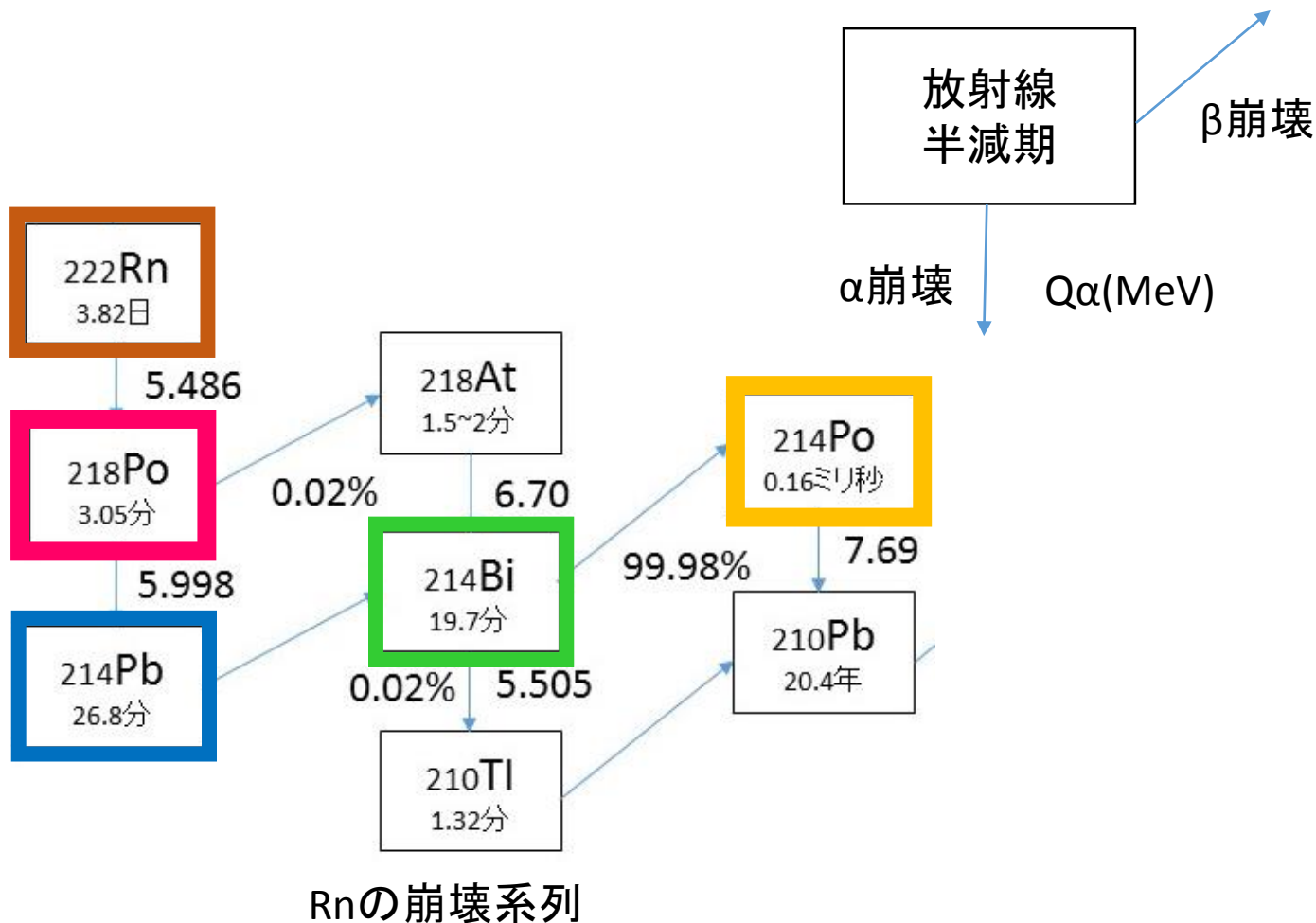
0MeV, 241Am, 218Po, 214Poの点をプロットし、0MeVと241Amを用いて求めた校正直線を引いた図である。

218Poはほとんど直線上であるが、214Poは直線よりも左にずれた。

→214Poのみ測定結果のchが文献値の値よりも低い???

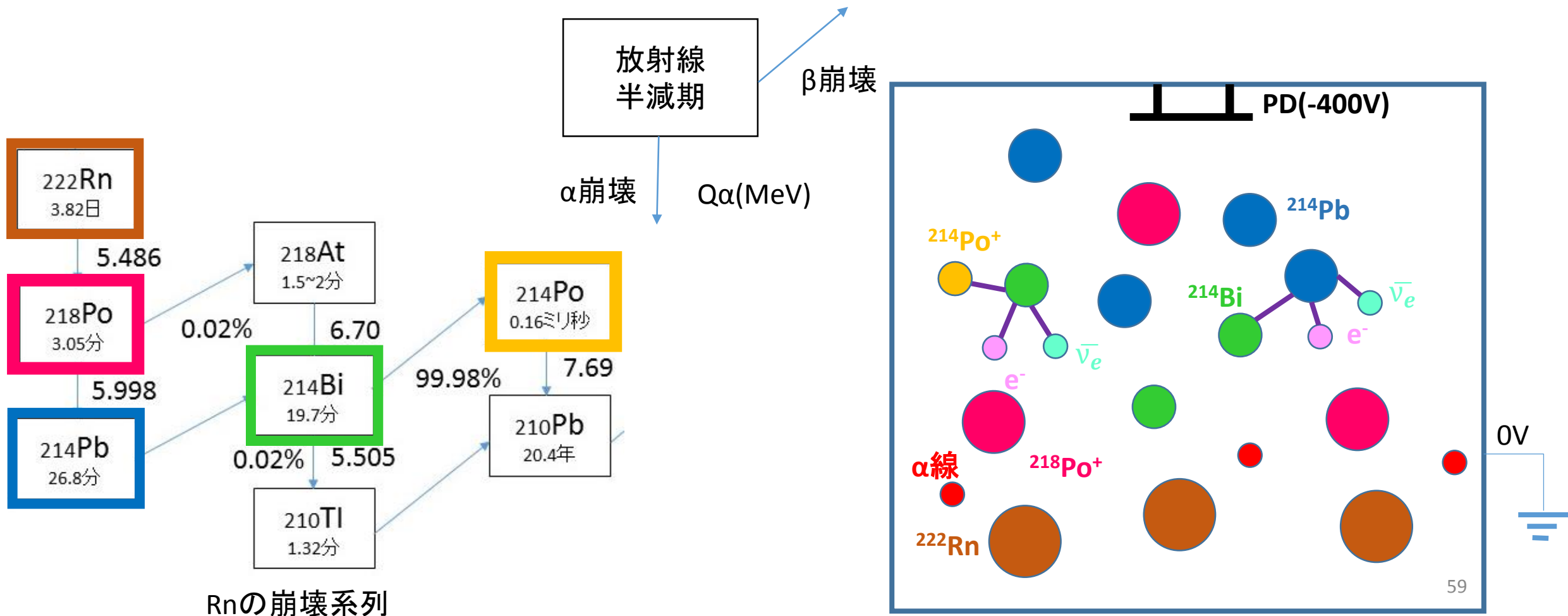
214Poのエネルギー損失

今回の実験では静電捕集法によってPD表面にPoを吸着させることでPoが α 崩壊したときの α 線のエネルギーを測定しているが、



214Poのエネルギー損失

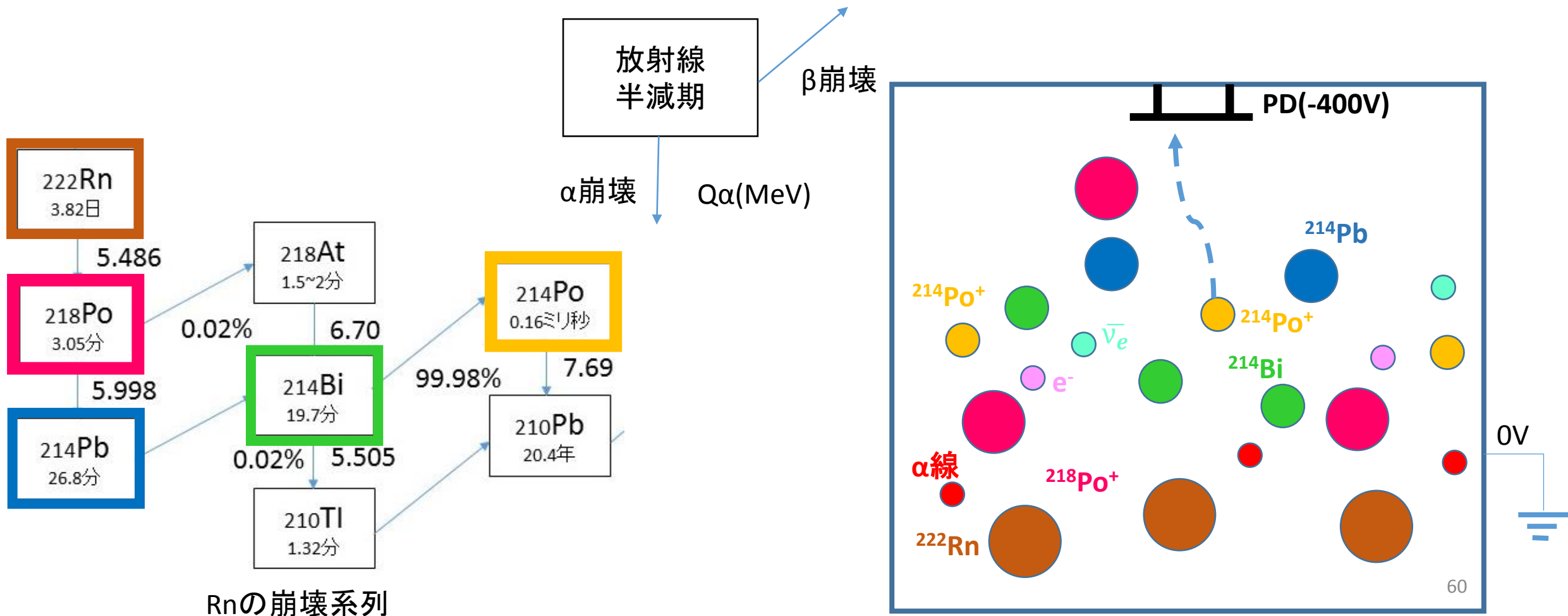
今回の実験では静電捕集法によってPD表面にPoを吸着させることでPoが α 崩壊したときの α 線のエネルギーを測定しているが、
静電捕集法でPD表面に吸着していた218Poが崩壊し、214Pb,214Biに崩壊しているときにPDへの吸着がはがれ、



214Poのエネルギー損失

今回の実験では静電捕集法によってPD表面にPoを吸着させることでPoが α 崩壊したときの α 線のエネルギーを測定しているが、

静電捕集法でPD表面に吸着していた ^{218}Po が崩壊し、 ^{214}Pb , ^{214}Bi に崩壊しているときにPDへの吸着がはがれ、 ^{214}Po に崩壊してまたPD表面に引き付けられるが、



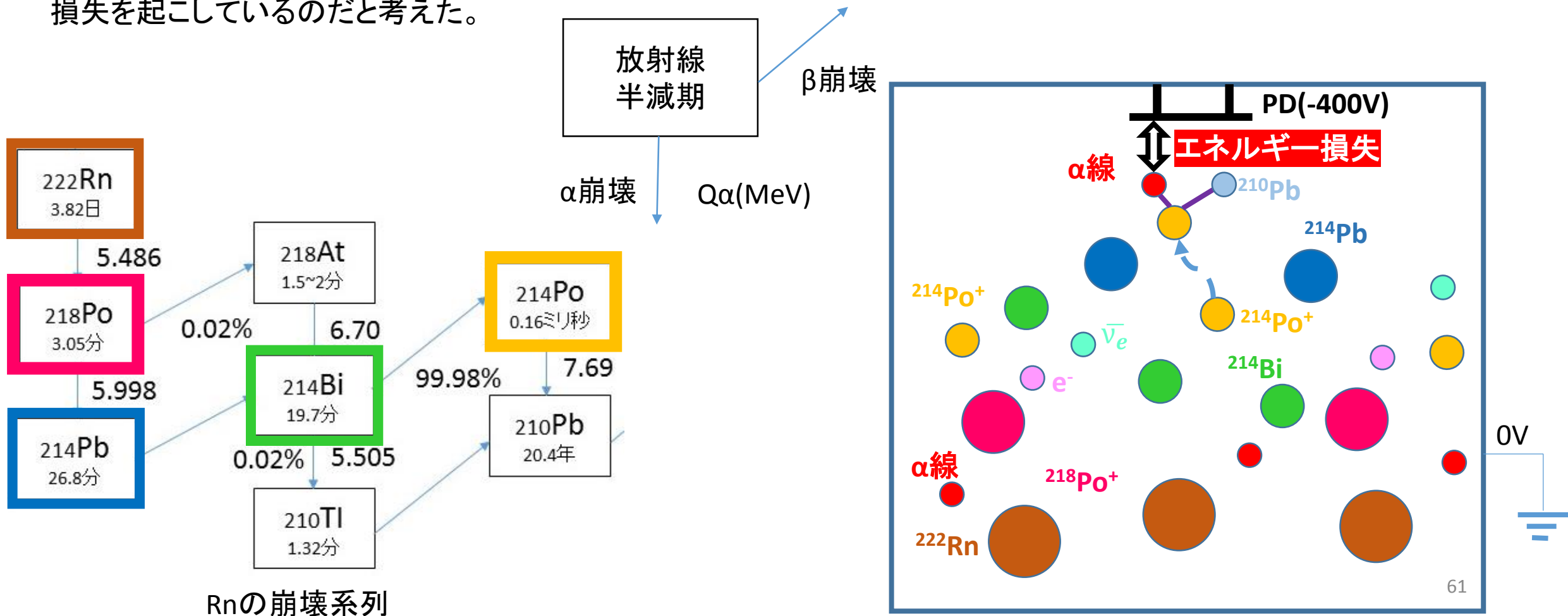
214Poのエネルギー損失

今回の実験では静電捕集法によってPD表面にPoを吸着させることでPoが α 崩壊したときの α 線のエネルギーを測定しているが、

静電捕集法でPD表面に吸着していた218Poが崩壊し、214Pb,214Biに崩壊しているときにPDへの吸着がはがれ

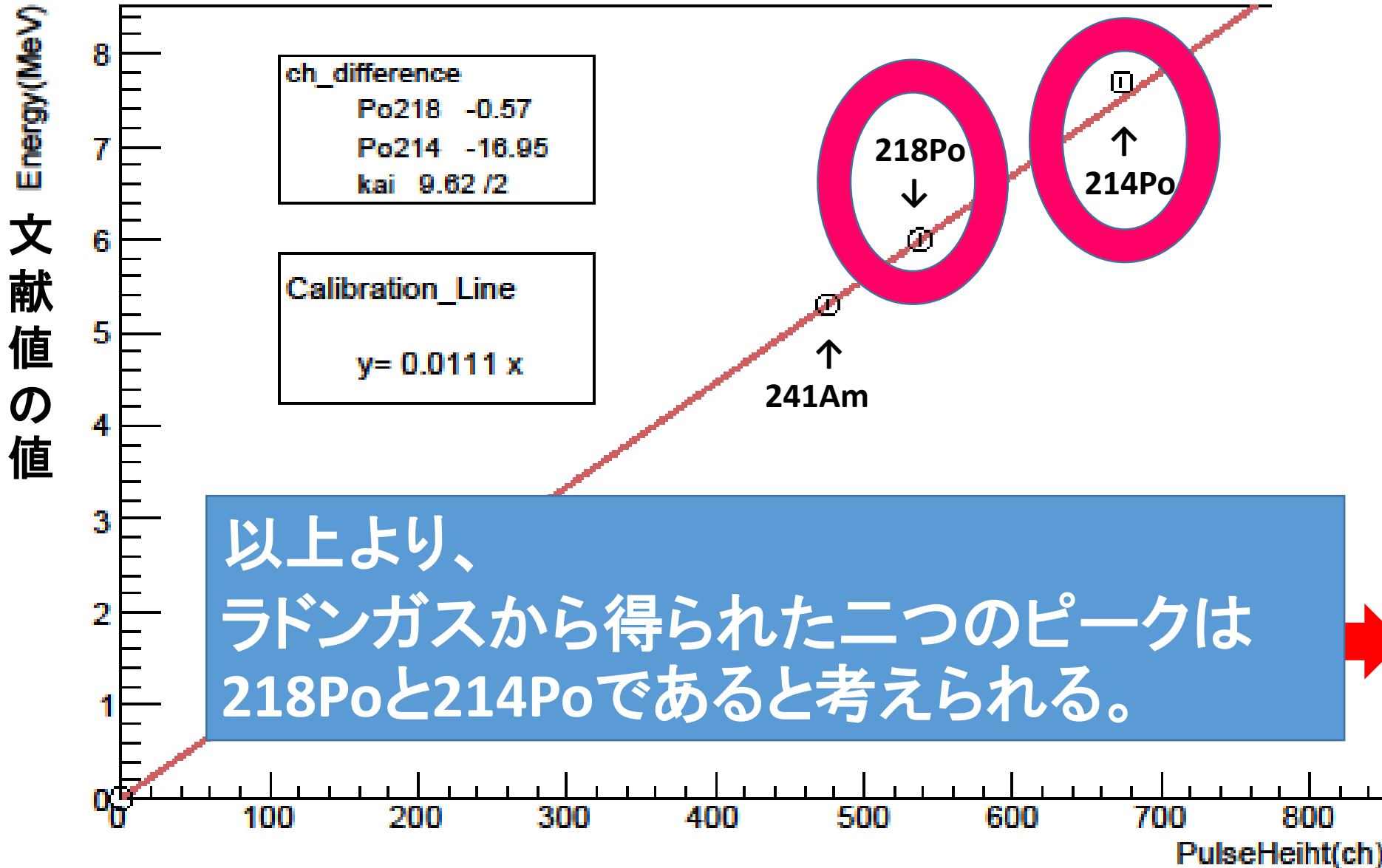
214Poに崩壊してまたPD表面に引き付けられるが、

214Poの半減期はとても短いため、PD表面に向かっている段階で α 崩壊し、PD表面との間の空気によってエネルギー損失を起こしているのだと考えた。



Energy-PulseHeight

Am校正直線との比較



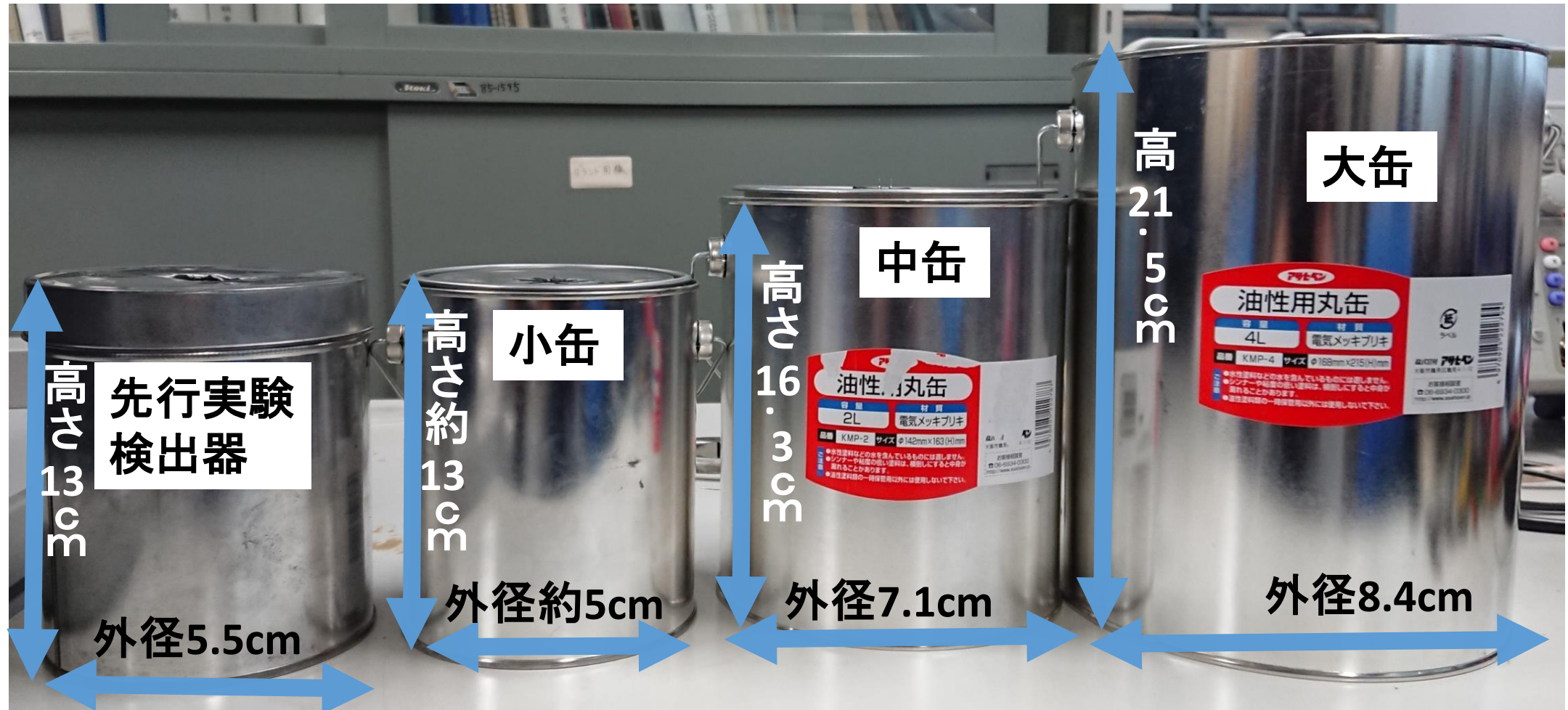
文献値の値

以上より、ラドンガスから得られた二つのピークは218Poと214Poであると考えられる。

次はそれらの半減期を求め

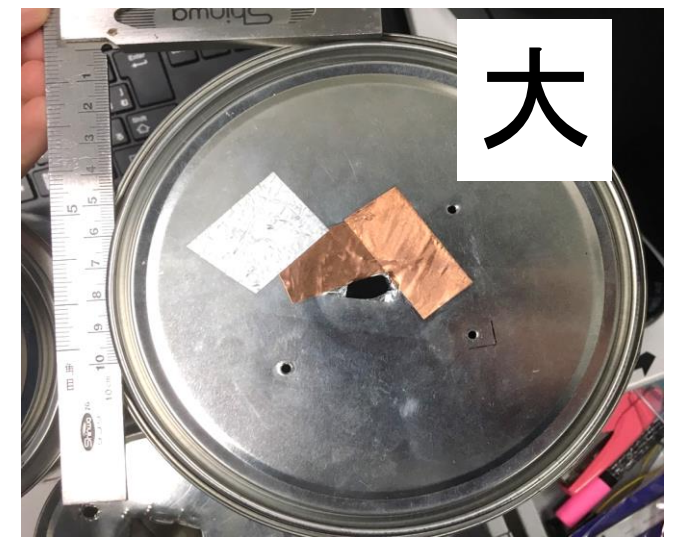
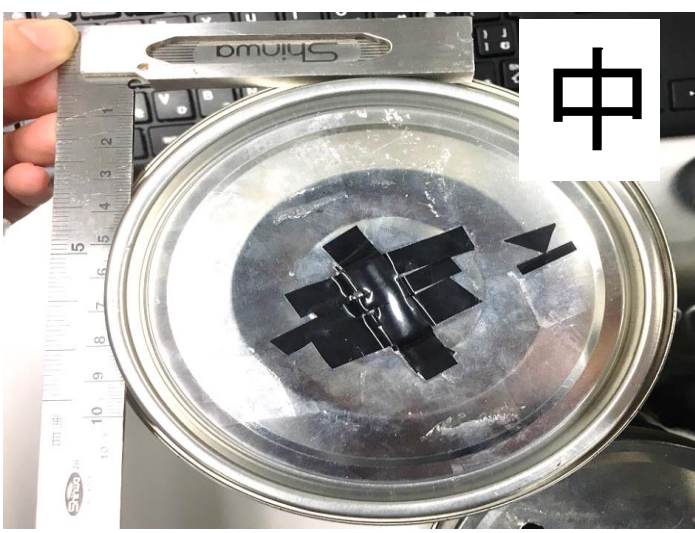
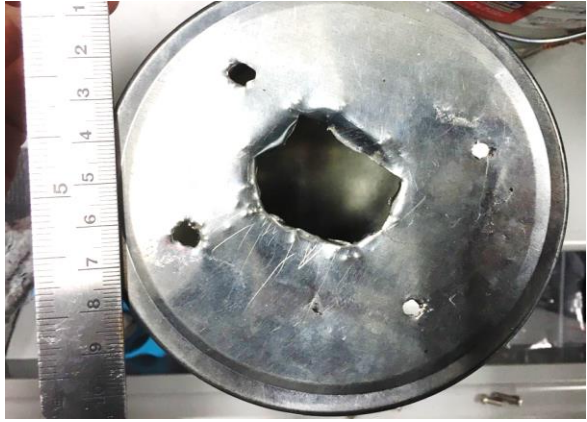
実験の測定結果

缶のサイズ

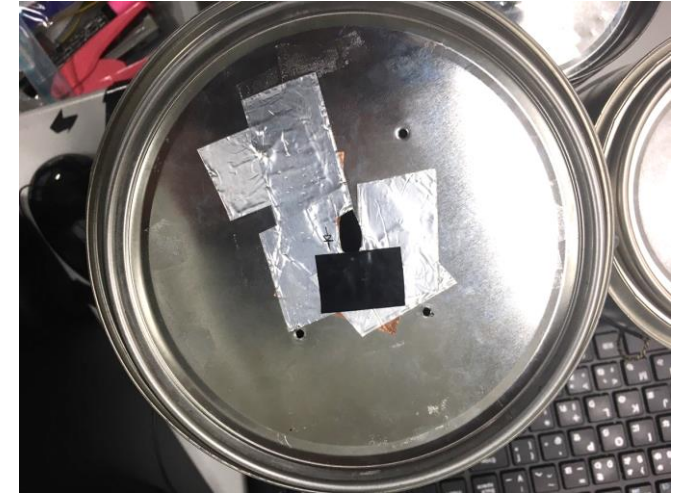
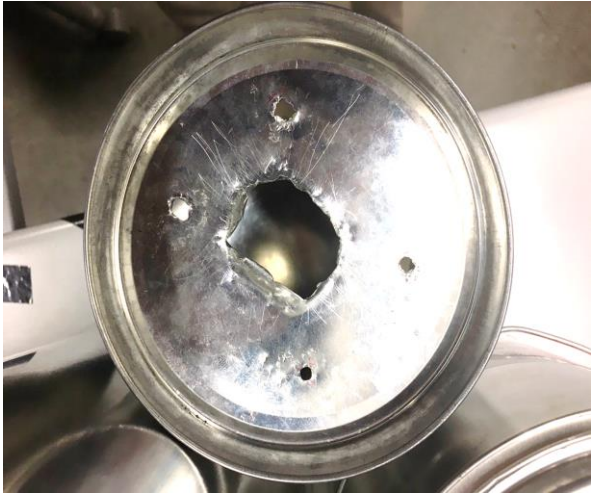


先行実験検出器

蓋の表

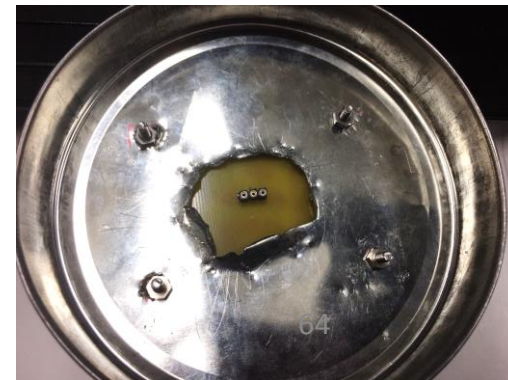
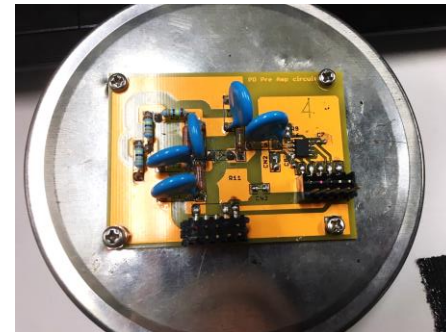


蓋の裏



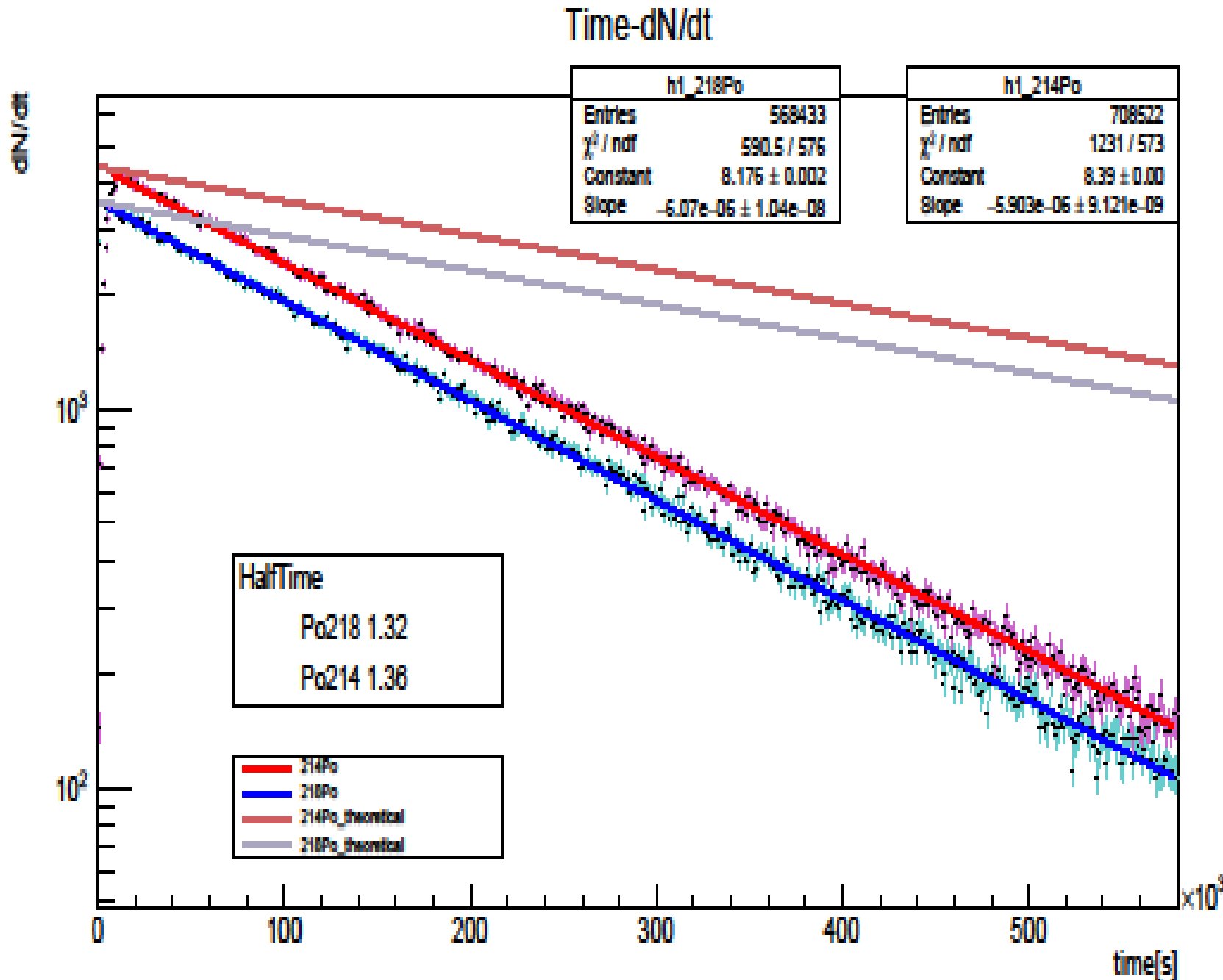
小缶が一番穴が開いている
密封度が高い順に並べると
中缶 > 大缶 > 小缶

プリアンプ
装着時→



長時間測定データを用いた考察

大缶にDOLL STONEを約72h入れての測定。
 ※測定一日目にシェイパーアンプが外れていたため、ラドン石を抜いて二日目からの測定である。
 1binあたり1000sである。



長時間測定データを用いた考察

測定時間による比較

大缶でのラドンガス測定データを用いての考察。
 上はfitする時間を表しており、横はbin数を表している。
 Fitする時間を長くすればするほど求められる半減期が短くなっている...?
 →測定初めのデータを含めているので最初の方はちゃんとfitができていないと考えた。
 24時間から測定時間を延ばしてもあまり大きな変化は見られないので
 測定時間は24時間が妥当だと考えた。

Bin数	時間	3時間 10800(s)	4時間 14400(s)	5時間 18000(s)	10時間 36000(s)	24時間 86400(s)	48時間 172800(s)	72時間 259200(s)	161時間 579600(s)
400	半減期T T[day]	エラー	5.63日	3.89日	1.59日	1.36日	1.34日	1.34日	1.32日
	崩壊定数 $\lambda[\times 10^{-6}]$	+6.272	-1.425	-2.062	-5.034	-5.88	-5.983	-5.98	-6.054
80	半減期T T[day]	エラー	2.23日	2.23日	1.51日	1.36日	1.34日	1.34日	1.33日
	崩壊定数 $\lambda[\times 10^{-6}]$		-3.59	-3.59	-5.3	-5.92	-5.992	-5.989	-6.05
40	半減期T T[day]	エラー	エラー	エラー	1.91日	1.35日	1.34日	1.34日	1.33日
	崩壊定数 $\lambda[\times 10^{-6}]$				-4.192	-5.936	-5.997	-5.99	-6.05 ₆₆

fitの場所による比較

下の表は20時間ごとにfitした時の半減期結果を表したものである。
 時間が経過するにつれて半減期が短くなるわけではないことが分かった。

長時間測定データを用いた考察

Bin数	時間	0~20時間	20~40時間	40~60時間	60~80時間	80~100時間	100~120時間	120~140時間	140~160時間
580 1binあたり 1000[s]	半減期T T[day]	1.39	1.34	1.33	1.39	1.21	1.28	1.38	1.85
	崩壊定数 $\lambda[\times 10^{-6}]$	5.779	6.004	6.01	5.767	6.621	6.261	5.825	4.334
290 1binあたり 2000[s]	半減期T T[day]	1.38	1.34	1.34	1.39	1.21	1.28	1.38	1.85
	崩壊定数 $\lambda[\times 10^{-6}]$	5.833	6.005	6	5.757	6.644	6.27	5.816	4.341
145 1binあたり 4000[s]	半減期T T[day]	1.37	1.34	1.34	1.39	1.20	1.28	1.38	1.83
	崩壊定数 $\lambda[\times 10^{-6}]$	5.867	6.002	5.992	5.759	6.685	6.244	5.809	4.396
58 1binあたり 10000[s]	半減期T T[day]	1.37	1.34	1.34	1.4	1.22	1.32	1.35	1.61
	崩壊定数 $\lambda[\times 10^{-6}]$	5.849	5.975	5.997	5.713	6.577	6.082	5.926	4.975 ₆₇

Arduinoのプログラム

```
const byte ADC_PIN = 0;
const byte TRG = 0; // 0: pin-2  1: pin-3
const byte bitmask = B00011111;

volatile int num; // データを入れる箱をつくる
int nloop = 100;

void setup(){

  pinMode(7,OUTPUT); // GATEピン
  digitalWrite(7,LOW); // 最初ゼロの値を与える
  attachInterrupt(TRG,GATE,RISING);
  // TRGがRisingしたらGATE関数を実行
  Serial.begin(9600);

  // speed test
  int time = millis();
  for( int i=0; i<nloop; i++){
    GATE();
  }
  Serial.println("// Speed Test //");
  Serial.print("loop of ");
```

```
Serial.println(nloop);
time = millis() - time;
Serial.print("Time: ");
Serial.print(time);
Serial.println(" ms");
}
```

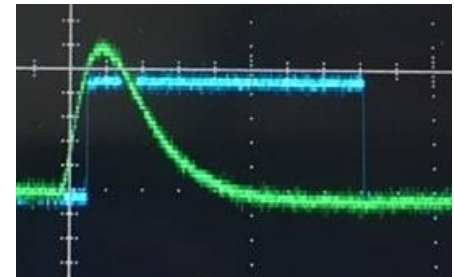
```
void GATE(){

  detachInterrupt(TRG);
  // not to interrupt during this function

  // Gate信号
  digitalWrite(7,HIGH); // 7をON
  delayMicroseconds(8); // 単位はusec
  digitalWrite(7,LOW); // 7をOFF

  num = analogRead(ADC_PIN);
  // アナログピンの値を読み込む

  delayMicroseconds(200); // 時間をおく
```



```

//リセット用のGate信号
digitalWrite(7,HIGH);
delayMicroseconds(3);
digitalWrite(7,LOW);

senddata( num );
attachInterrupt(TRG,GATE,RISING);
}
byte getParity( byte val ){

  byte x;
  x = val >> 4;
  val = val^x;
  x = val >> 2;
  val = val^x;
  x = val >> 1;
  val = val^x;

  return ( val & 1 );
}

void senddata( int val ){

  //convert data

```

```

byte lowVal = ( val & bitmask );
byte higVal = ( ( val >> 5 ) & bitmask );

//horizontal parity check
byte lowParity = getParity( lowVal );
byte higParity = getParity( higVal );

//vertical parity check
byte andParity = getParity( higVal &
lowVal );
byte orParity = getParity( higVal |
lowVal );

//make packet
byte higBit = B10000000;
byte lowBit = B00000000;
higBit += ( higVal << 2 );
lowBit += ( lowVal << 2 );
higBit += ( andParity << 1 );
lowBit += ( orParity << 1 );
higBit += higParity;
lowBit += lowParity;

Serial.write( higBit );

```

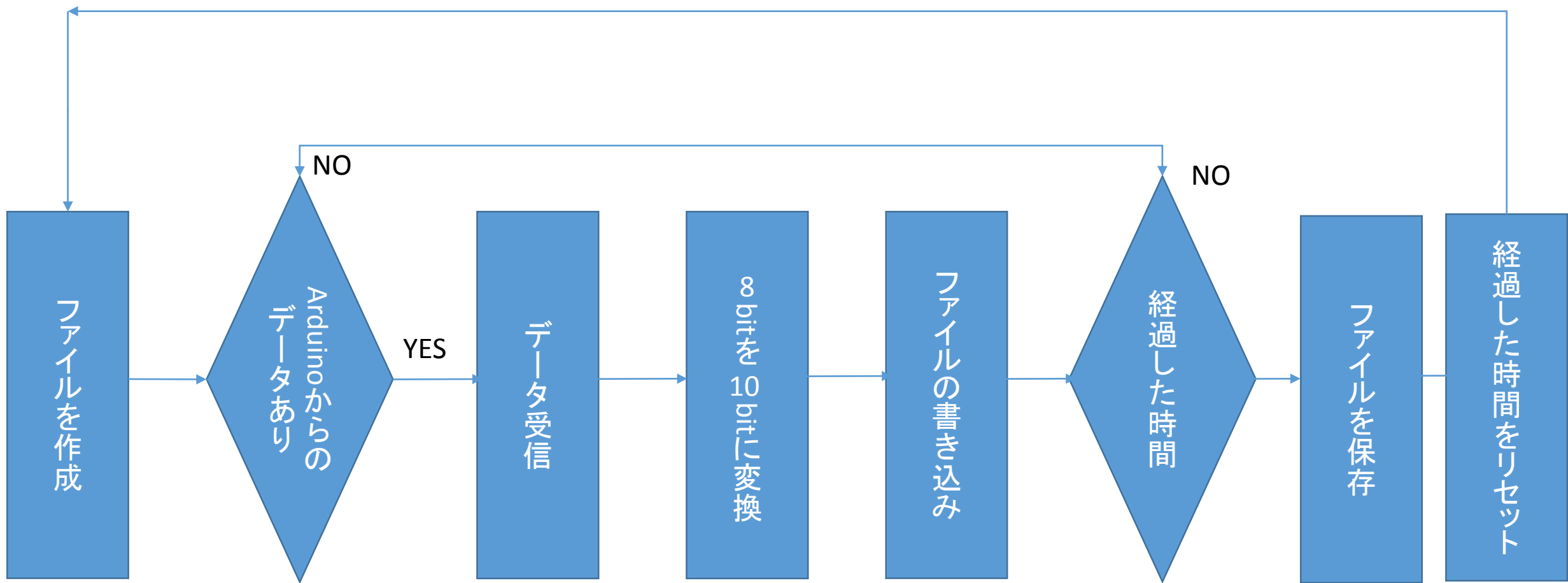
```

Serial.write( lowBit );

//byte tmp[2];
//tmp[0] = higBit;
//tmp[1] = lowBit;
//Serial.write( tmp, 2 );
}

void loop(){
  //delay(3000);
  //senddata( 750 );
}

```



Processingのフローチャート

```

#include<fstream>
using namespace std;//fstream使うときのおまじない
void hist_mean_hakoari () {
//ヒストグラムの作成TH1F * h1 = new TH1F("name", "title", bin数, 下限, 上限);
TH1F * h1 = new TH1F("h1", "h1", 1024, 0, 1023);
//ファイルの読み込み
ifstream ifs("20181111_19h_radonishinasi.dat");
double x;
while (ifs >> x) {
h1->Fill(x);
}
ifs.close();
//bin数をn個で1個に換算 Rebin(n):bin数1/n
h1->Rebin(4);
//ヒストグラムを描く
h1->Draw();
//関数の定義 TF1* f1 = new TF1("name", "式");
TF1* f1 = new TF1("gaus", "gaus(0)");
TF1* f2 = new TF1("gaus", "gaus(0)");
TF1* f3 = new TF1("gaus", "gaus(0)");
//凡例の表示 TLegend * tg = new TLegend(x1,y1,x2,y2,"title");
TLegend * tg1 = new TLegend(0.2, 0.7, 0.4, 0.85, "peak1");
TLegend * tg2 = new TLegend(0.2, 0.5, 0.4, 0.65, "peak2");
TLegend * tg3 = new TLegend(0.2, 0.15, 0.4, 0.25, "");
TLegend * tg4 = new TLegend(0.2, 0.3, 0.4, 0.45, "peak3");
//ピークの線の色指定
f1->SetLineColor(9);
f2->SetLineColor(8);
f3->SetLineColor(6);
h1->Fit(f1, "+", "", 605, 630); //Fitting
h1->Fit(f2, "+", "", 665, 685);
h1->Fit(f3, "+", "", 755, 785);

```

```

//GetParameter(); 0:ピークのy, 1:ピークのx, 2:sigma
Double_t p0 = f1->GetParameter(1);
Double_t p1 = f2->GetParameter(1);
Double_t p2 = f1->GetParameter(2);
Double_t p3 = f2->GetParameter(2);
Double_t p6 = f3->GetParameter(1);
Double_t p7 = f3->GetParameter(2);
//GerParError(n);n番目のパラメータのエラー
Double_t p4 = f1->GetParError(1);
Double_t p5 = f2->GetParError(1);
Double_t p8 = f3->GetParError(1);
//凡例の定義 (どこに何を表示するか)
tg1->AddEntry((TObject*)0, Form("Mean %.2f", p0), "");
tg1->AddEntry((TObject*)0, Form("Sigma %.2f", p2), "");
tg1->AddEntry((TObject*)0, Form("MeanError %.2f", p4), "");
tg2->AddEntry((TObject*)0, Form("Mean %.2f", p1), "");
tg2->AddEntry((TObject*)0, Form("Sigma %.2f", p3), "");
tg2->AddEntry((TObject*)0, Form("MeanError %.2f", p5), "");
tg4->AddEntry((TObject*)0, Form("Mean %.2f", p6), "");
tg4->AddEntry((TObject*)0, Form("Sigma %.2f", p7), "");
tg4->AddEntry((TObject*)0, Form("MeanError %.2f", p8), "");
//tg->AddEntry(fit関数, "name", "l"); "l"はLineのl
tg3->AddEntry(f1, "peak1", "l");
tg3->AddEntry(f2, "peak2", "l");
tg3->AddEntry(f3, "peak3", "l");
//凡例の表示
tg1->Draw();
tg2->Draw();
tg3->Draw();
tg4->Draw();
return;
}

```

ヒストグラムの プログラム


```

#include<fstream>
using namespace std;//fstream使うときのおまじない
void time_dNdt() {
//canvasの作成 TCanvas * c1 = new TCanvas("name", "title", 横の長さ, 縦の長さ);
TCanvas * c1 = new TCanvas("c1", "20181111_19h_radonishinasi", 600, 400);
//ヒストグラムの作成 TH1F * h1 = new TH1F("name", "title", bin数, 下限, 上限);
TH1F * h1 = new TH1F("218Po", "Time - dN / dt; time[s]; dN / dt", 500, 0, 70000);
TH1F * h2 = new TH1F("214Po", "20181111_19h_radonishinasi", 500, 0, 70000);
//配列の定義
int time, ch;
double p1mean, p2mean, p1sigma, p2sigma, p1min, p1max, p2min, p2max;
//peakの情報入力
p1mean = 617.28;
p2mean = 770.72;
p1sigma = 6.31;
p2sigma = 5.89;
p1min = p1mean - p1sigma * 2.0;
p1max = p1mean + p1sigma * 2.0;
p2min = p2mean - p2sigma * 2.0;
p2max = p2mean + p2sigma * 2.0;
//ファイルの読み込み
ifstream ifs("20181111_19h_radonishinasi_time_ch.dat");
while (ifs >> time >> ch) {
if (ch >= p1min && ch <= p1max) { //peak1の範囲のchであれば
h1->Fill(time);//timeの箱で取ったカウント数のデータをヒストグラムに入れる
}
if (ch >= p2min && ch <= p2max) { //peak2の範囲のchであれば
h2->Fill(time);//timeの箱で取ったカウント数のデータをヒストグラムに入れる
}
}
}
ifs.close();

```

time-dN/dtの プログラム

```

//統計ボックスの表示/非表示
h1->SetStats(1);//表示
//ヒストグラムの線の色指定
h1->SetLineColor(9);
h2->SetLineColor(6);
//bin数をn個で1個に換算 Rebin(n):bin数1/n
h1->Rebin(4);
h2->Rebin(4);
//ヒストグラムを描く
h1->Draw();
h2->Draw("same");
//統計Boxの位置を変える 場所はDraw()の後
c1->Update();//paintする
TPaveStats *st1 = (TPaveStats*)h1->FindObject("stats");
st1->SetTextColor(9);
TPaveStats *st2 = (TPaveStats*)h2->FindObject("stats");
st2->SetTextColor(6);
return;
}

```

```

#include<fstream>
using namespace std;//fstream使うときのおまじない
void time_dNdt_fit() {
//canvasの作成 TCanvas * c1 = new TCanvas("name", "title", 横の長さ, 縦の長さ);
TCanvas * c1 = new TCanvas("c1", "20181111_19h_radonishinasi", 600, 400);
//ヒストグラムの作成TH1F * h1 = new TH1F("name", "title;Xtitle;Ytitle", bin数, 下限, 上限);
TH1F * h1 = new TH1F("218Po", "Time-dN/dt;time[s];dN/dt", 400, 0, 70000);
TH1F * h2 = new TH1F("214Po", "h2;time[s];dN/dt", 400, 0, 70000);
//配列の定義
int time, ch;
double p1mean, p2mean, p1sigma, p2sigma, p1min, p1max, p2min, p2max;
//peakの情報入力
p1mean = 617.28;
p2mean = 770.72;
p1sigma = 6.31;
p2sigma = 5.89;
p1min = p1mean - p1sigma * 2;
p1max = p1mean + p1sigma * 2;
p2min = p2mean - p2sigma * 2;
p2max = p2mean + p2sigma * 2;
//ファイルの読み込み
ifstream ifs("20181111_19h_radonishinasi_time_ch.dat");
while (ifs >> time >> ch) {
if (ch >= p1min && ch <= p1max) { //peak1の範囲のchであれば
h1->Fill(time); //timeの箱で取ったカウント数のデータをヒストグラムに入れる
}
if (ch >= p2min && ch <= p2max) { //peak2の範囲のchであれば
h2->Fill(time); //timeの箱で取ったカウント数のデータをヒストグラムに入れる
}
}
}
ifs.close();

```

time-dN/dt fitの プログラム

```

//ヒストグラムの線の色指定
h1->SetLineColor(9);
h2->SetLineColor(6);
//bin数をn個で1個に換算 Rebin(n):bin数1/n
h1->Rebin(10);
h2->Rebin(10);
//軸の対数表示
c1->SetLogy();
//c1->SetLogy(false); //元に戻す
//統計ボックスの表示/非表示
h1->SetStats(1); //表示
//ヒストグラムを描く
//"e"はエラーバーの表示、"same"は重ね書き、"sames"で統計ボックスも
重ね書き
h1->Draw("e");
h2->Draw("sames e");
//統計Boxの位置を変える
c1->Update(); //paintする
TPaveStats *st1 = (TPaveStats*)h1->FindObject("stats");
TPaveStats *st2 = (TPaveStats*)h2->FindObject("stats");
st1->SetX1NDC(0.2); //x座標の始点
st1->SetX2NDC(0.4); //x座標の終点
st1->SetY1NDC(0.3); //y座標の始点
st1->SetY2NDC(0.5); //y座標の終点
st2->SetX1NDC(0.6); //x座標の始点
st2->SetX2NDC(0.8); //x座標の終点
st2->SetY1NDC(0.55); //y座標の始点
st2->SetY2NDC(0.75); //y座標の終点
c1->Modified(); //オブジェクトの設定変更を伝える
//統計Boxの設定 Fit情報の表示
gStyle->SetOptFit();

```

```

//統計Boxの設定 各統計情報の表示/非表示
gStyle->SetOptStat(1000000011);
//関数の定義 TF1* f1 = new TF1("name", "式");
TF1* f1 = new TF1("expo", "expo"); //exp(p0+p1*x)
TF1* f2 = new TF1("expo", "expo"); //exp(p0+p1*x)
//凡例の表示 TLegend * tg = new TLegend(x1,y1,x2,y2,"title");
TLegend * tg3 = new TLegend(0.2, 0.15, 0.4, 0.25, "");
//Fit線の色指定
f1->SetLineColor(4);
f2->SetLineColor(2);
//Fitting
h1->Fit(f1, "+", "", 0, 70000);
h2->Fit(f2, "+", "", 6000, 70000);
//y切片の値を214PoのFitに合わせた時の文献値直線
TF1 * f3 = new TF1("f3", "exp(7.004-2.09e-6*x)", 0, 70000);
f3->SetNpx(500);
f3->SetLineColor(46);
f3->Draw("same");
//y切片の値を218PoのFitに合わせた時の文献値直線
TF1 * f4 = new TF1("f4", "exp(6.594-2.09e-6*x)", 0, 70000);
f4->SetNpx(500);
f4->SetLineColor(40);
f4->Draw("same");
//tg->AddEntry(fit関数, "name", "l"); "l"はLineのl
tg3->AddEntry(f2, "214Po", "l");
tg3->AddEntry(f1, "218Po", "l");
tg3->AddEntry(f3, "214Po_theoretical", "l");
tg3->AddEntry(f4, "218Po_theoretical", "l");
//凡例の表示
tg3->Draw();
return;
}

```

較正直線の プログラム

```
#include<fstream>
using namespace std;//fstream使うときのおまじない
void MeV_ch() {
//canvasの作成 TCanvas * c1 = new TCanvas("name", "title", 横の長さ, 縦の長さ);
TCanvas * c1 = new TCanvas("c1", "Am_ch-counts", 600, 400);
//ヒストグラムの作成TH1F * h1 = new TH1F("name", "title", bin数, 下限, 上限);
TH1F * h1 = new TH1F("Am", "0MeV_Am", 1024, 0, 1023);
TH1F * h2 = new TH1F("0MeV", "0MeV_Am; PulseHeight[ch]; counts", 1024, 0, 1023);
TH1F * h3 = new TH1F("Po", "0MeV_Am; PulseHeight[ch]; counts", 1024, 0, 1023);
//ファイルの読み込み
ifstream ifs("20181120_5m_Am_16_time_ch.dat");
double x1, y1;
while (ifs >> x1 >> y1) {
h1->Fill(y1);
}
ifs.close();
ifstream ifs2("20181120_5m_Am_0ch_12_time_ch.dat");
double x2, y2;
while (ifs2 >> x2 >> y2) {
h2->Fill(y2);
}
ifs2.close();
ifstream ifs3("20181111_19h_radonishinasi_time_ch.dat");
double x3, y3;
while (ifs3 >> x3 >> y3) {
h3->Fill(y3);
}
ifs3.close();
//関数の定義 TF1* f1 = new TF1("name", "式");
TF1* f1 = new TF1("gaus", "gaus(0)");
TF1* f2 = new TF1("gaus", "gaus(0)");
TF1* f4 = new TF1("gaus", "gaus(0)");
```

```
TF1* f5 = new TF1("gaus", "gaus(0)");
//ピークの線の色指定
f1->SetLineColor(9);
f2->SetLineColor(6);
//Fitting
h3->Fit(f4, "+", "", 605, 625);
h3->Fit(f5, "+", "", 755, 785);
h1->Fit(f1, "+", "", 540, 565);
h2->Fit(f2, "+", "", 20, 50);
//ヒストグラムを描く
h2->Draw("");
h1->Draw("sames");
//統計Boxの位置を変える 場所はDraw()の後
c1->Update();//paintする
TPaveStats *st1 = (TPaveStats*)h1->FindObject("stats");
st1->SetTextColor(9);
TPaveStats *st2 = (TPaveStats*)h2->FindObject("stats");
st2->SetTextColor(6);
st2->SetY1NDC(0.575);//y座標の始点
st2->SetY2NDC(0.735);//y座標の終点
c1->Modified();//オブジェクトの設定変更を伝える
```

```

//凡例の表示 TLegend * tg = new TLegend(x1,y1,x2,y2,"title");
TLegend * tg1 = new TLegend(0.78, 0.375, 0.98, 0.535, "");
//FitParameterの取得
//GetParameter(); 0:ピークのy, 1:ピークのx, 2:sigma
Double_t p0 = f1->GetParameter(1);
Double_t p1 = f2->GetParameter(1);
Double_t p2 = f4->GetParameter(1);
Double_t p3 = f5->GetParameter(1);
Double_t p4 = f1->GetParError(1);
Double_t p5 = f2->GetParError(1);
Double_t p6 = f4->GetParError(1);
Double_t p7 = f5->GetParError(1);
//凡例の定義
tg1->AddEntry(f1, "Am", "l");
tg1->AddEntry(f2, "0MeV", "l");
//凡例の表示
tg1->Draw();
//canvasの作成 TCanvas * c1 = new TCanvas("name", "title", 横の長さ, 縦の長さ);
TCanvas * c2 = new TCanvas("c2", "Energy_PulseHeight", 600, 400);
//配列の定義
double Am_x, zero_x, Am_y, zero_y, dx,
dy, dx1, dy1, Po218_y, Po214_y, dx2, dy2;
double Po214_x, Po218_x;
Am_x = 5.4;
zero_x = 0.0;
Am_y = p0;
zero_y = p1;
dx = Am_x - zero_x;
dy = Am_y - zero_y;
Po218_x = 5.998;
Po214_x = 7.69;

```

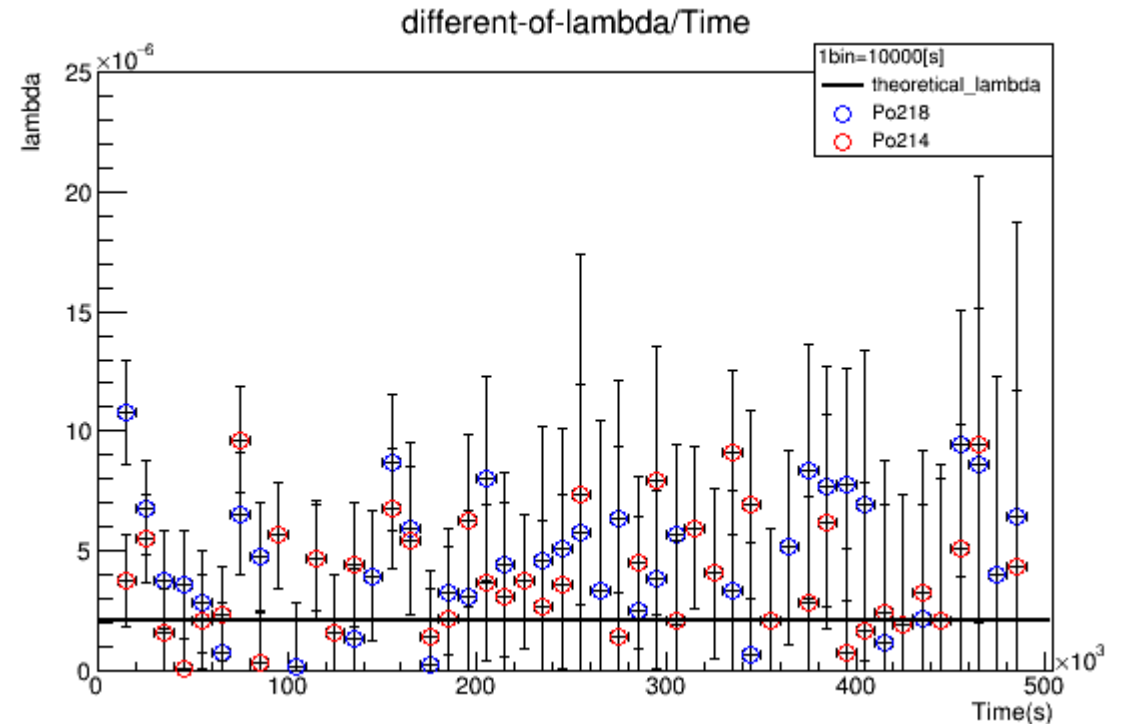
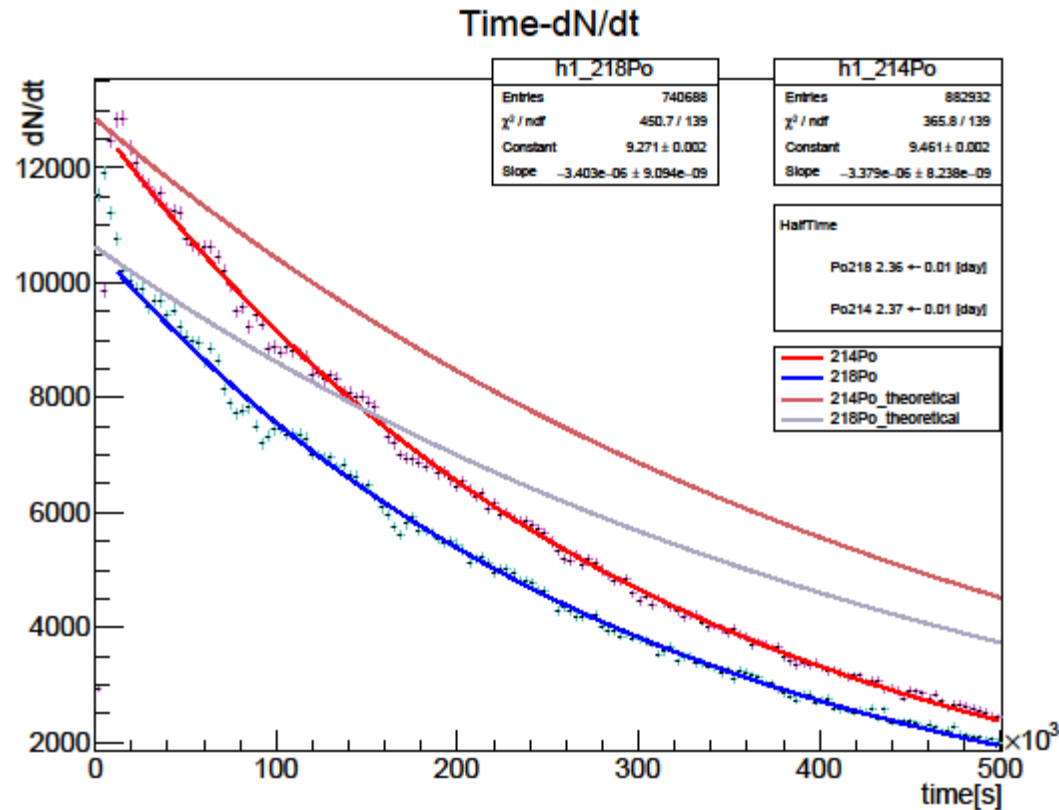
```

dx1 = Po218_x - zero_x;
dy1 = p2 - p1;
dx2 = Po214_x - zero_x;
dy2 = p3 - p1;
//配列の定義
Double_t x[4] = { 0.0, 5.4, Po218_x, Po214_x };
Double_t y[4] = { p1, p0, p2, p3 };
Double_t xe[4] = { 0.0, 0.0, 0.0, 0.0 };
Double_t ye[4] = { p5, p4, p6, p7 };
//各点にプロット
TGraphErrors * tgr1 = new TGraphErrors(4, x, y, xe, ye);
tgr1->SetTitle("Energy-PulseHeight;Energy (MeV);PulseHeight (ch)");
tgr1->SetMarkerStyle(1);
tgr1->SetMarkerColor(1);
tgr1->SetMarkerSize(1);
tgr1->Draw("AP");
//較正直線
TF1 * f3 = new TF1("f3", "[0]/[1]*x+[2]", 0, 8.5);
f3->SetParameters(dy, dx, zero_y);
f3->SetNpx(500); //500個の点でプロット
f3->SetLineColor(46);
f3->Draw("same");

return;
}

```

長時間測定データの解析結果



1月8日のラドン石を8日入れて140時間測定したときの時の測定結果である。

左は縦軸を対数表示にしていなかったときの測定結果で、右は時間ごとにfitした時の崩壊定数の大きさをプロットしたものである。

左の測定結果を見ると文献値の減り方よりも早く減っていることが分かり、時間がたつにつれて文献値の減り方とfitした結果の差があまり増えなくなるので割合的に減っていくと考えました。

減る理由としてはラドンガスの段階で缶から逃げているためだと考えました。

右の測定結果をより測定時間によって求められる半減期はあまり変わらないと考えました。