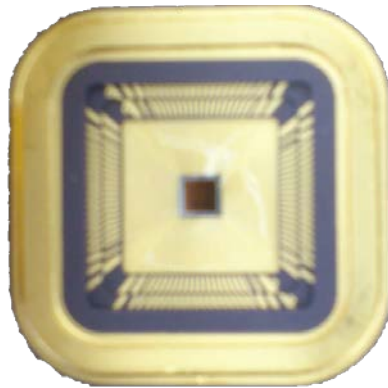


# マイコンを用いたXRPIX制御



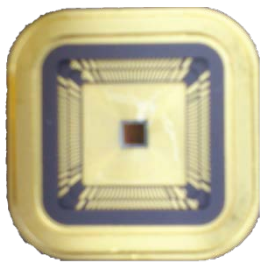
赤井隆一

京都大学理学部理学科

2013年度 卒業研究P6発表会

# 実験の目的

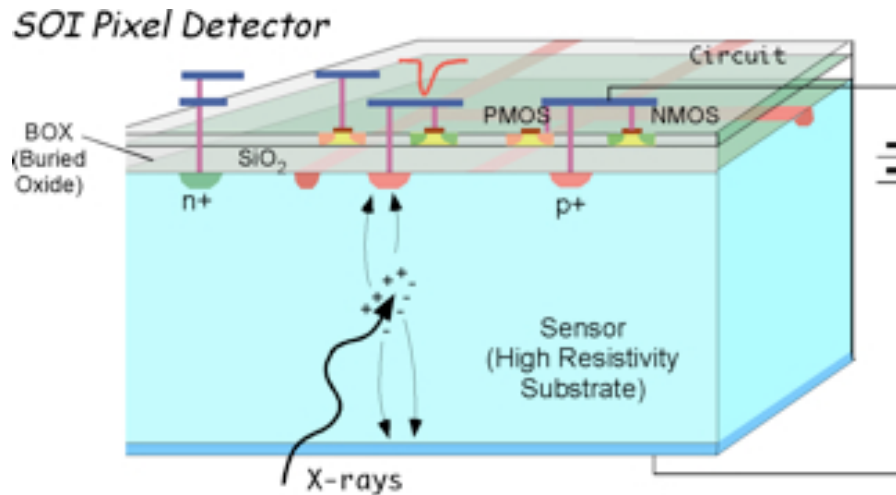
- FPGAを使用せず、マイコンでXRPIXに制御信号を入力しX線信号データを取得する。



# XRPIX

- 京大がKEK(高エネルギー加速器研究機構)と共同で設計開発したX線天文学用SOIPIX
- 各ピクセルにトリガー回路と読み出し回路を搭載したシリコンアクティブピクセルセンサー
- 反同時係数法を用いることにより、非X線バックグラウンド(NXB)の大幅な除去が可能

# SOIPIXとは



- SOI(Silicon-On-Insulator)技術を利用し比抵抗の異なるSiレイヤーを一体化
- 上部のSiレイヤーをSOI-CMOS回路、下部のSiレイヤーをセンサーとして利用
- 従来型のCMOSセンサーに比べ、回路部が高速で省電力、放射線耐性に優れる。

# 今回用いた素子の仕様

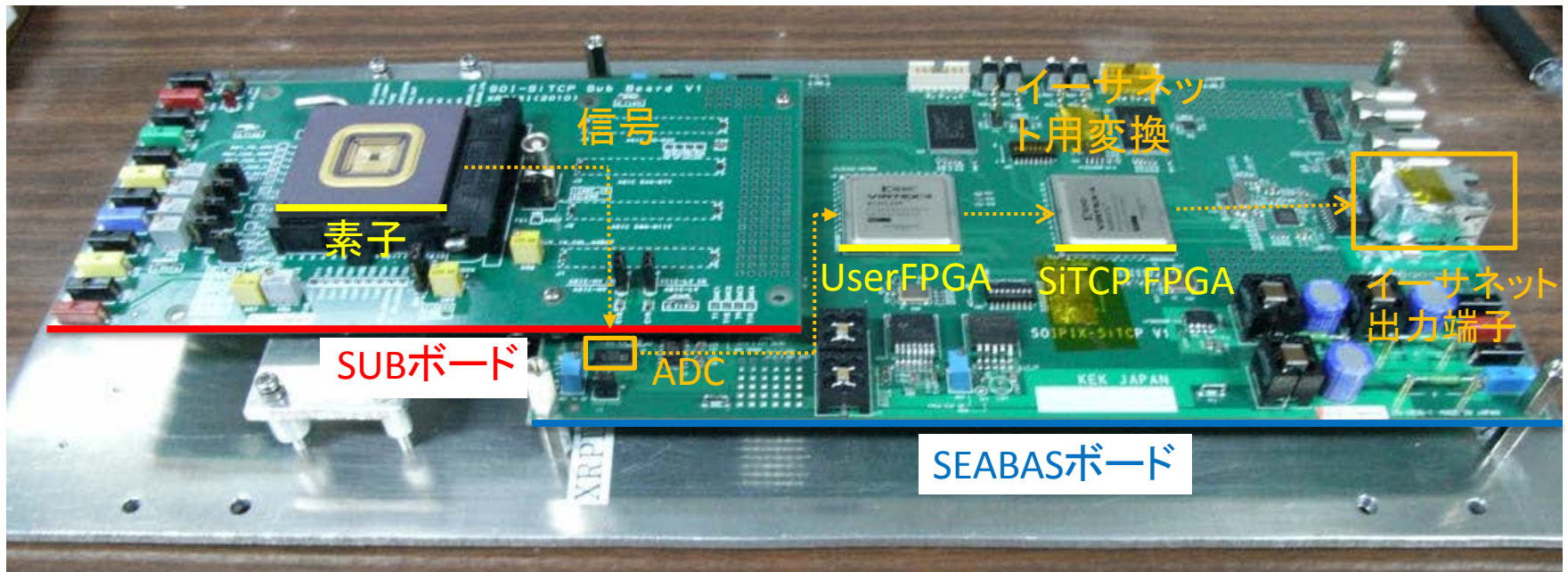
- 実験で用いた素子はXRPIX1b-CZ-100 $\mu$ m

## XRPIX1b-CZ-100 $\mu$ m仕様

ピクセルサイズ	30 $\mu$ m x 30 $\mu$ m
フォーマット	32 x 32 pixel
センサー層厚	100 $\mu$ m
センサー層の比抵抗	0.7 k $\Omega$ cm

# 信号の流れ

- 通常、XRPIX1bの制御にはFPGA(Field Programmable Gate Array)を用いる。



# FPGA

- ・ユーザーがハードウェア記述言語を用いて設計を書き込む事で、構成を変更可能な集積回路。

- ・ハードウェア記述言語にはVHDL, Verilog HDL等がある。



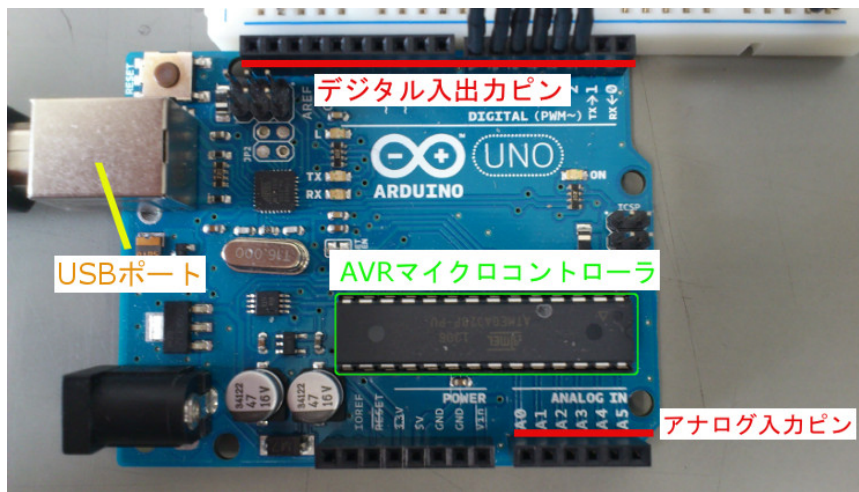


## マイクロコントローラ(マイコン)

- コンピュータの動作に必要な主要部品を一つのチップに実装したものの。
- あらかじめプログラムを書き込んでおくことで、独立した電子機器の制御が可能。
- 今回の実験ではAtmel AVR マイクロコントローラを搭載したArduino UNOを使用



# Arduino UNO 主な入出力端子

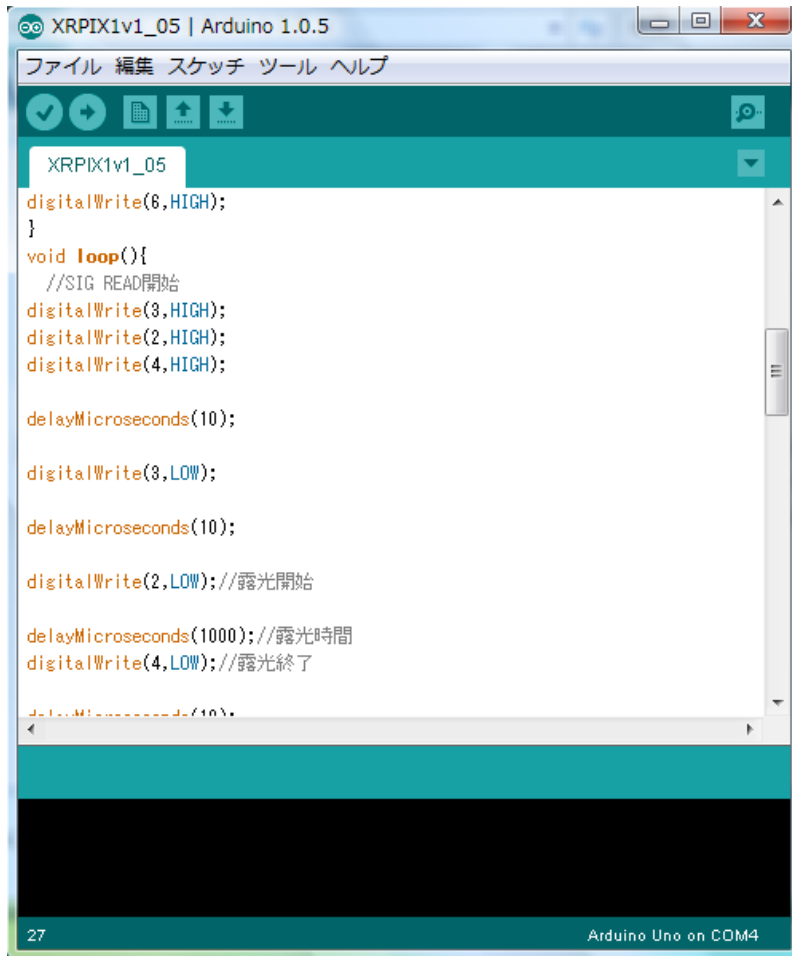


- ・14本のデジタル入出力ピン  
...5V(HIGH),0V(LOW)を出力可能

- ・6本のアナログ入力ピン  
...5Vまたは1.1Vを基準電圧として1024  
段階で電圧を測定可能  
今回は1.1Vを基準として使用  
(1単位1.1mv)

- ・USBポート  
...PCとの通信、マイコンへのプログラム  
書き込みに使用

# Arduino IDE(統合開発環境)



The screenshot shows the Arduino IDE window titled "XRPIX1v1\_05 | Arduino 1.0.5". The menu bar includes "ファイル", "編集", "スケッチ", "ツール", and "ヘルプ". The toolbar contains icons for saving, undo, redo, and running. The main editor area displays the following code:

```
XRPIX1v1_05
digitalWrite(6,HIGH);
}
void loop(){
  //SIG READ開始
  digitalWrite(3,HIGH);
  digitalWrite(2,HIGH);
  digitalWrite(4,HIGH);

  delayMicroseconds(10);

  digitalWrite(3,LOW);

  delayMicroseconds(10);

  digitalWrite(2,LOW);//露光開始

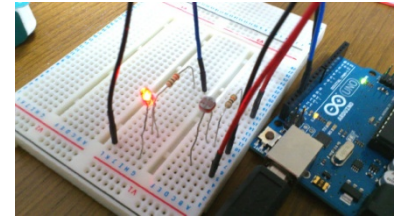
  delayMicroseconds(1000);//露光時間
  digitalWrite(4,LOW);//露光終了

  delayMicroseconds(10);
```

The status bar at the bottom indicates "27" and "Arduino Uno on COM4".

- C言語風の構文
- 作成したプログラムは gccによってコンパイルされhexファイルに変換後マイコンに書き込まれる

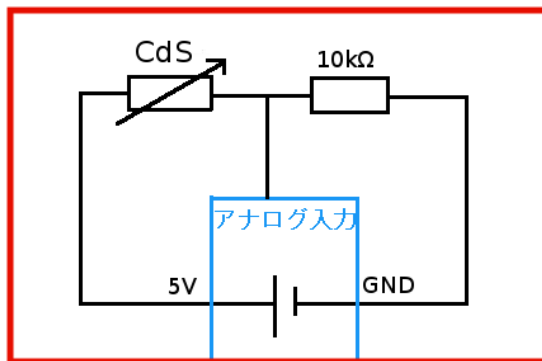
# マイコン使用例



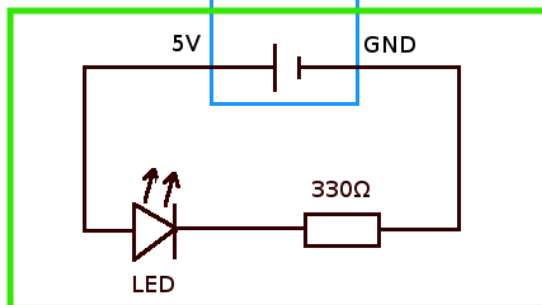
- 周囲の明るさに応じてon/offするLED

光を当てると抵抗の減少するCdSセルを光センサーとして利用

センサー回路



マイコン



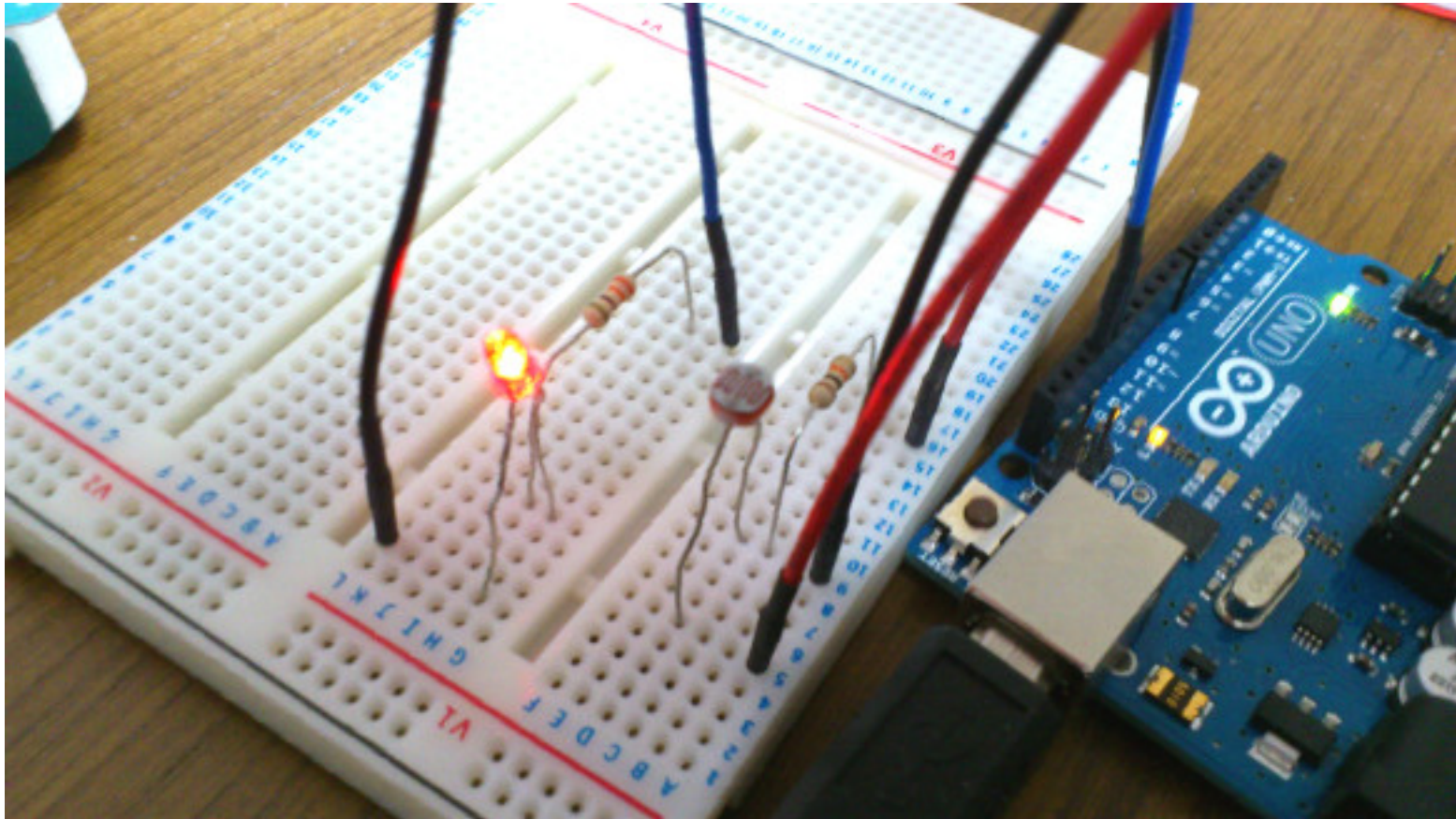
LED回路

CdSセルの抵抗の変化によりマイコンアナログ入力ピンにかかる電圧が変化

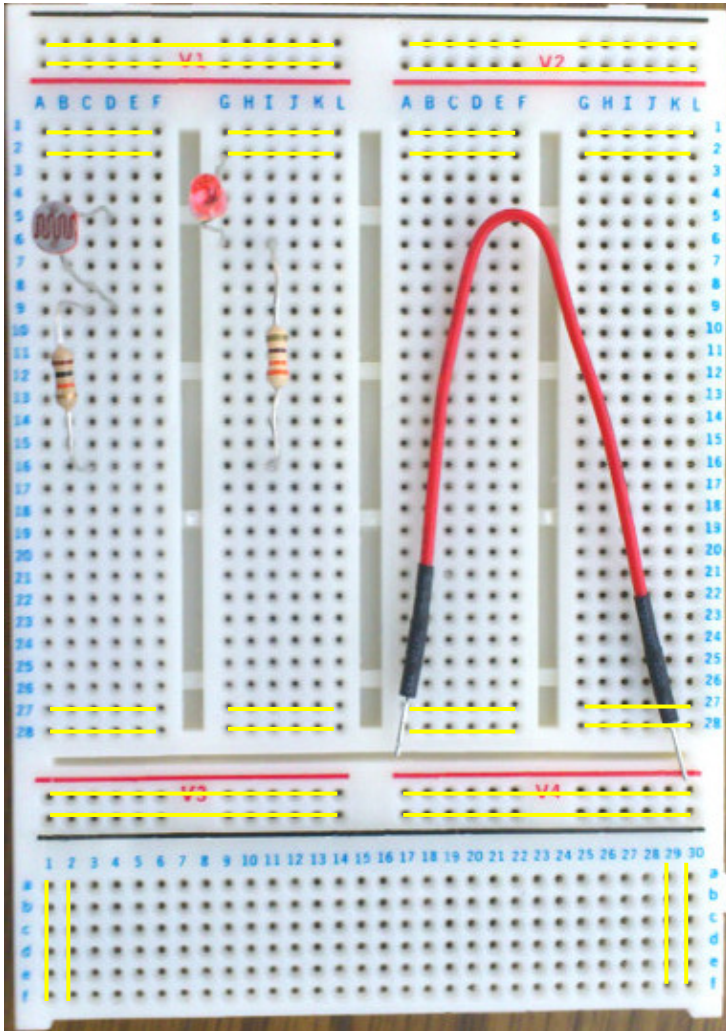


入力電圧がしきい値以下になるとLED回路に電圧を出力させる

# 実際の回路



# ソルダーレス・ブレッドボード



- 抵抗等の部品とジャンプワイヤを差し込む事で簡単に回路を構成、組み換えができる試作用基板
- 図中のライン方向の穴は内部で電氣的に接続している

# マイコンのプログラミング

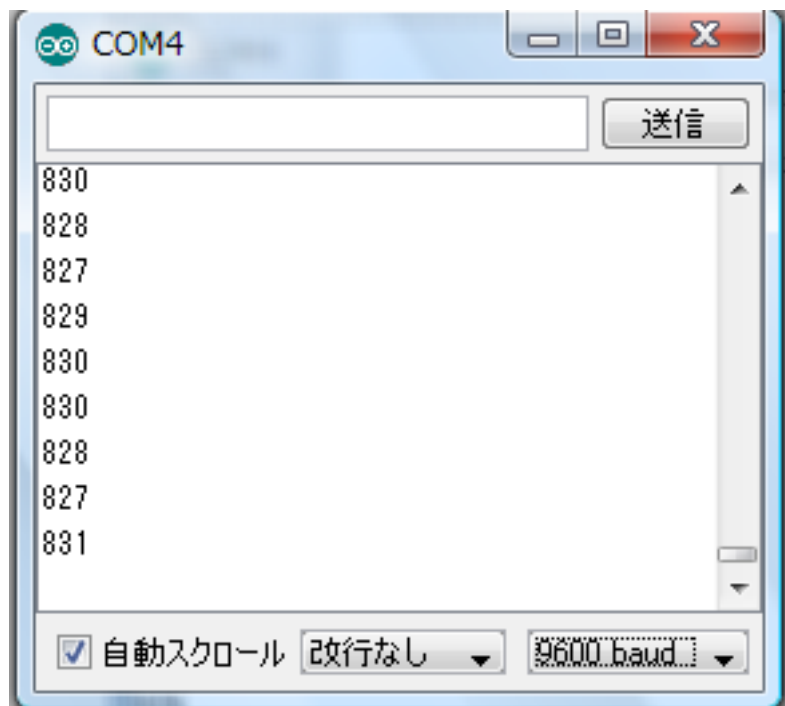
```
int i;//0-1023

void setup(){
  pinMode(13,OUTPUT); //デジタル13番を出力に設定
  Serial.begin(9600); //9600bpsでポートを開く
}

void loop(){
  i = analogRead(0); //0-1023
  if(i<500){ //500がしきい値
    digitalWrite(13,HIGH); //LEDオン
  }else{
    digitalWrite(13,LOW); //LEDオフ
  }
  Serial.print(i,DEC); //シリアル通信で電圧値を送信
  Serial.print("\n");
  delay(1000);
}
```



# PC側の受信データ(電圧値)

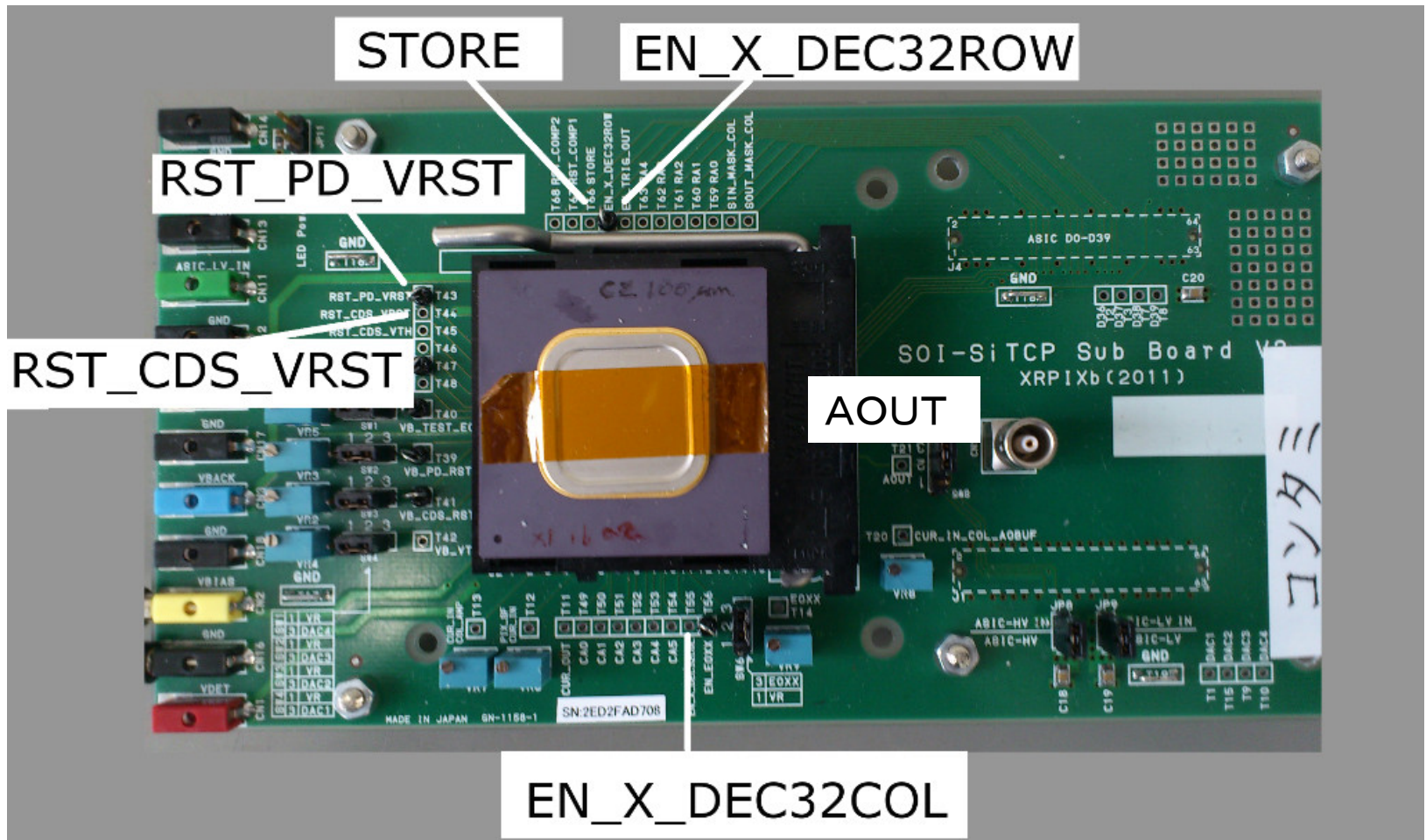


値を見てしきい値を設定  
すれば完成

Arduino IDE付属のシリアル  
モニタ

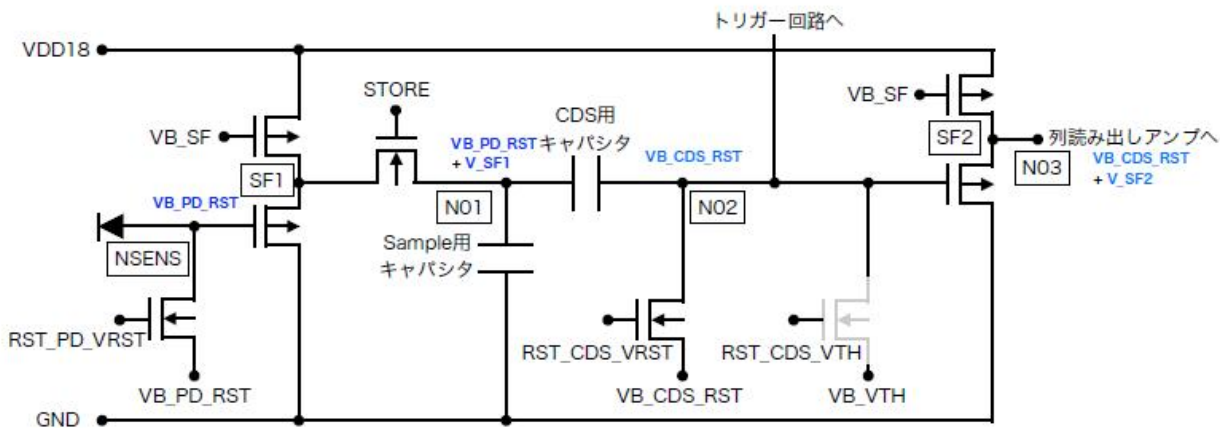
# XRPIX1bからの信号の読み出し

- 読み出しに使用する6つの信号の入出力場所



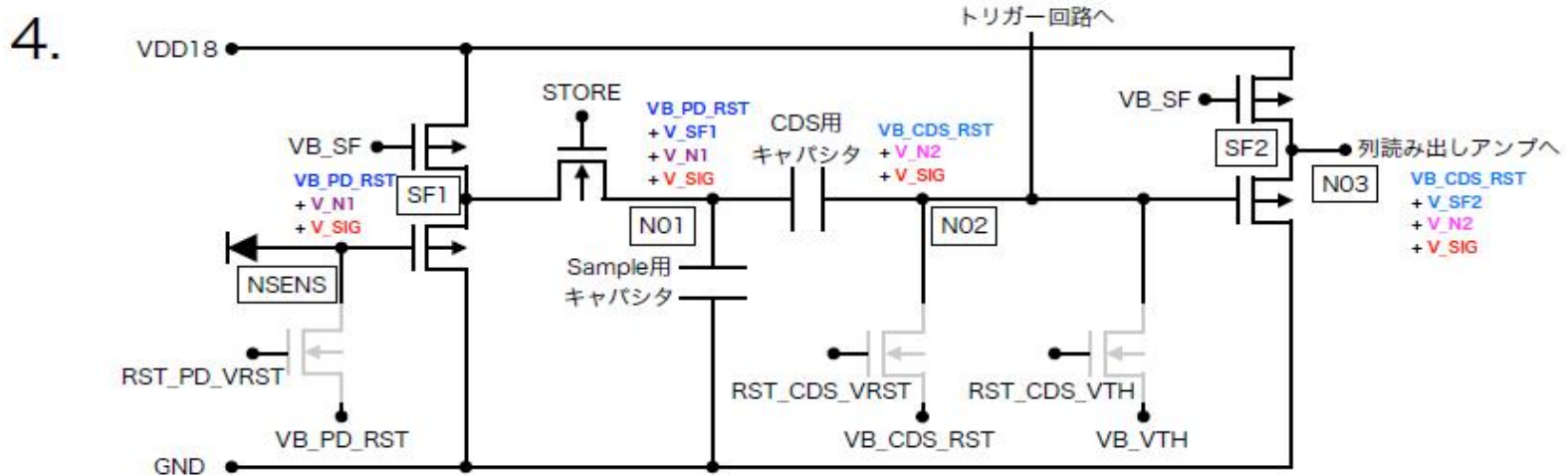


# ・各ピクセルの読み出し回路



- RST\_PD\_VRST... VSENSノードをリセット電圧に設定
- RST\_CDS\_VRST... N02をリセット電圧に設定
- STORE... オンにしている間、Sampleキャパシタに信号が積分される
- EN\_X\_DEC32ROW... ピクセルの位置指定(行)を有効にする
- EN\_X\_DEC32COL... ピクセルの位置指定(列)を有効にする

# X線入射後電圧読み出し手順



- 1.NSENSとN02をリセット電圧に固定
- 2.RST\_PD\_VRSTをオフ
- 3.RST\_CDS\_VRSTをオフ(露光開始,露光時間1ms)
- 4.N03の電圧( $V\_SIG + V\_N2S + V\_SF2$ )をアンプを通しAOUTから読み出す

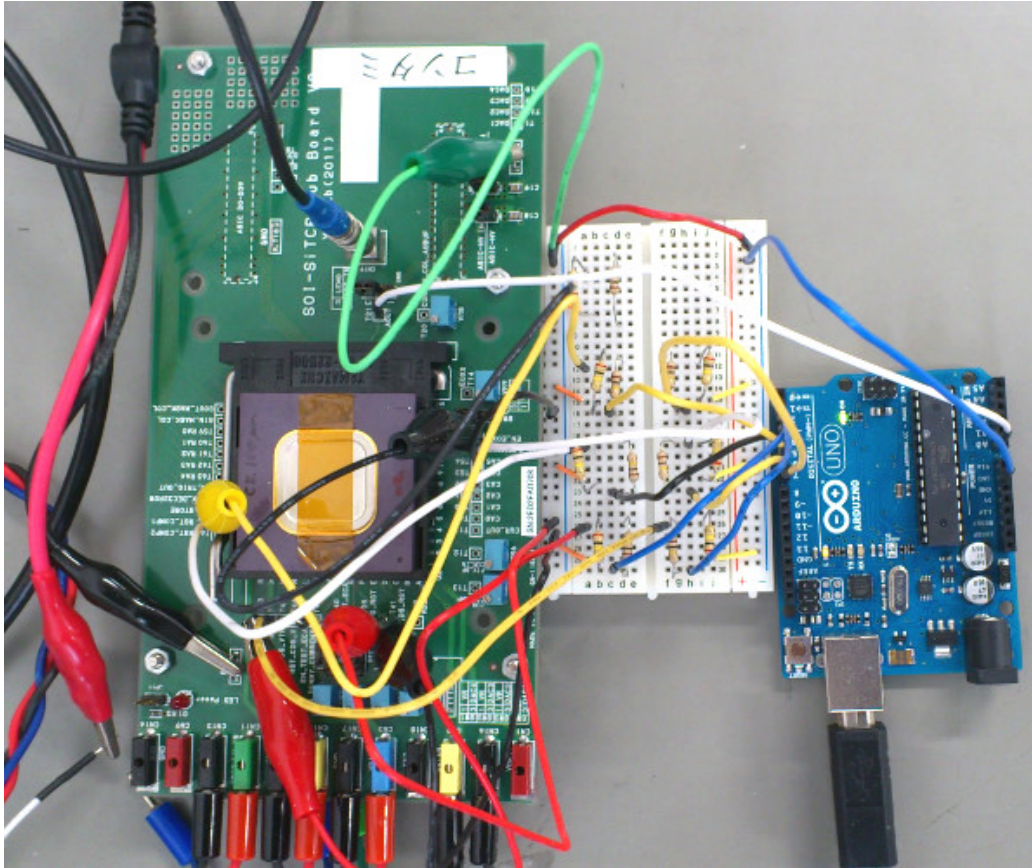
# リセット電圧読み出し

- 1.NSENSとN02をリセット電圧に固定
- 2.RST\_PD\_VRSTをオフ
- 3.RST\_CDS\_VRSTをオフ
- 4.N03の電圧( $V_{N2R}+V_{SF2}$ )をアンプを通しAOUTから読み出す

- X線入射後電圧ーリセット電圧

$$= \underbrace{V_{SIG}}_{\text{信号}} + \underbrace{(V_{N2S} - V_{N2R})}_{\text{ノイズ}}$$

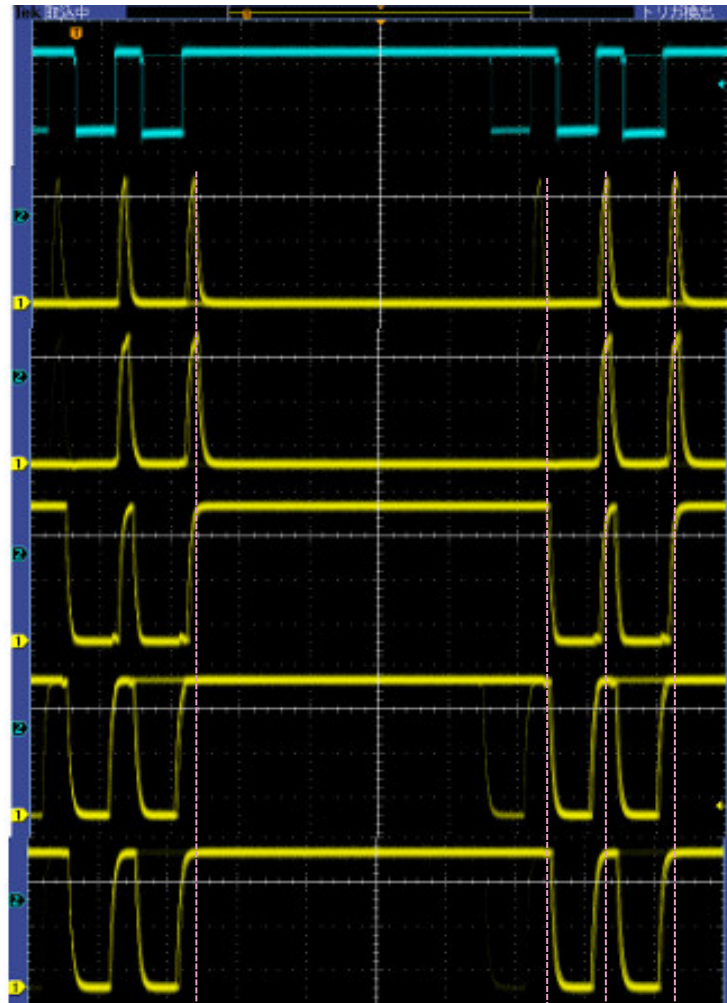
# 配線の様子



- マイコンからSUBボードへ5つの信号を出力
- SUBボードのAOUT(アナログ信号出力)をマイコンのアナログ入力ピンに接続
- 間のブレッドボードは電圧調整の為の分圧回路

# 信号のタイミング

- AOUT
- RST\_PD\_VRST
- RST\_CDS\_VRST
- STORE
- EN\_X\_DEC32ROW
- EN\_X\_DEC32COL



1ms

露光  
期間

X線電圧  
読み出し  
100µs

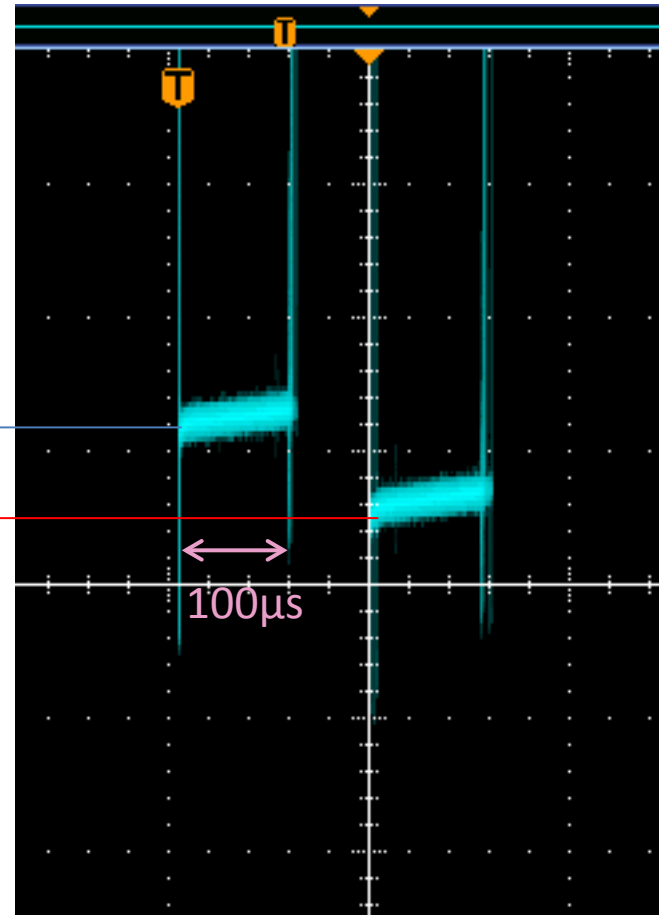
リセット  
読み出し  
100µs

# 信号の波形

信号電圧値

X線入射後電圧

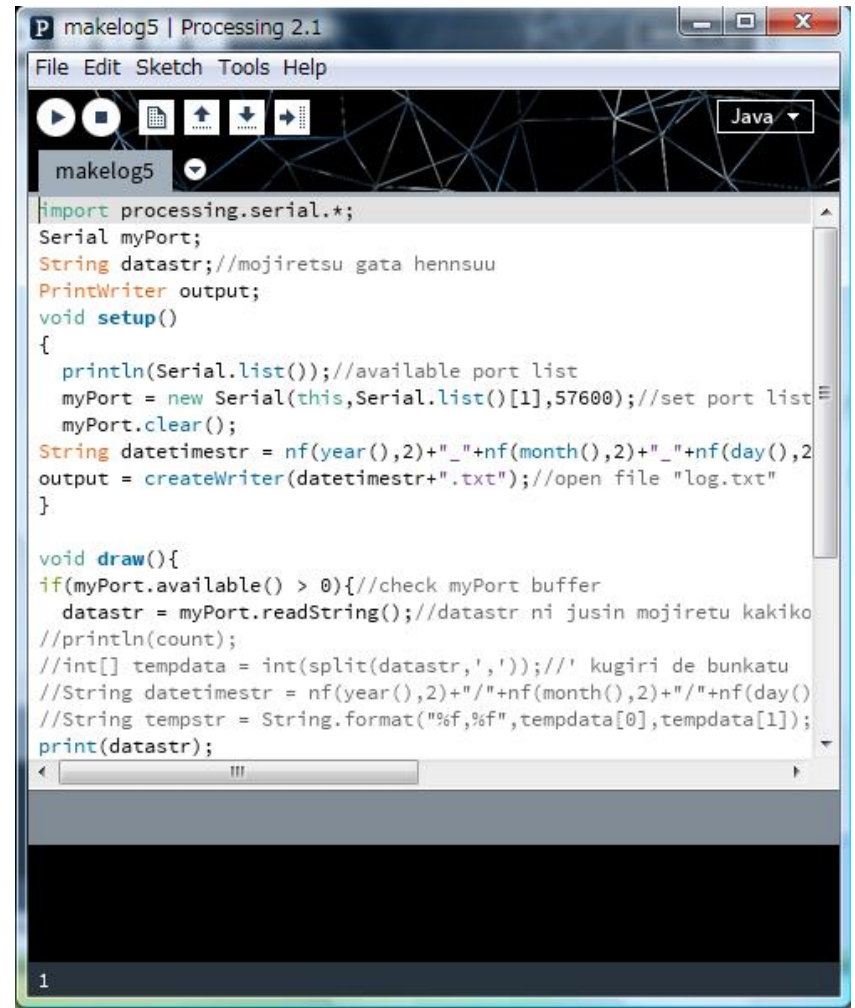
リセット電圧



マイコンのアナログ入力ピンを使って各電圧を読み出し、差を計算してPCへ送信する

# PC側のデータの記録方法

- Processingによるシリアル通信記録プログラム

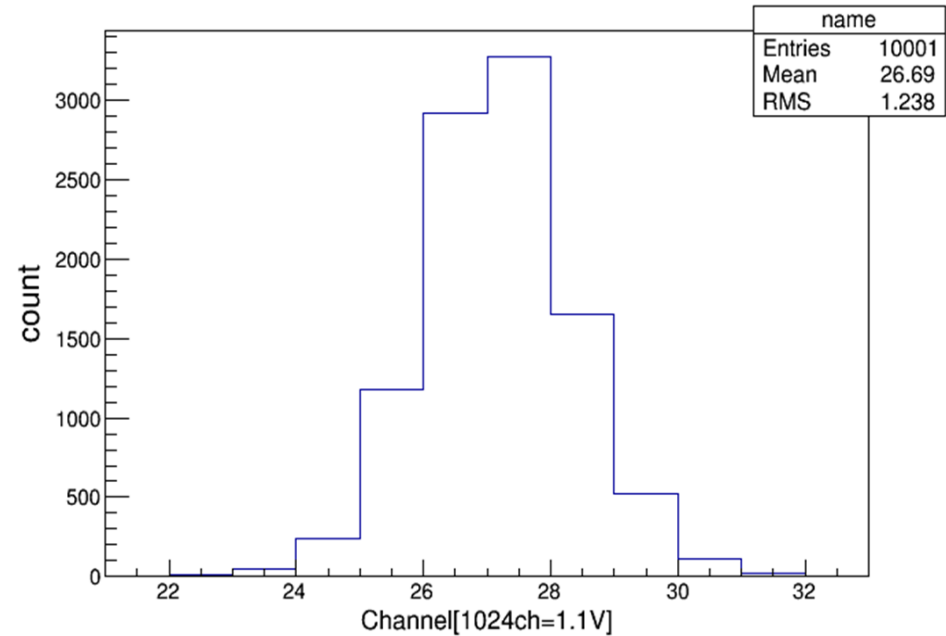
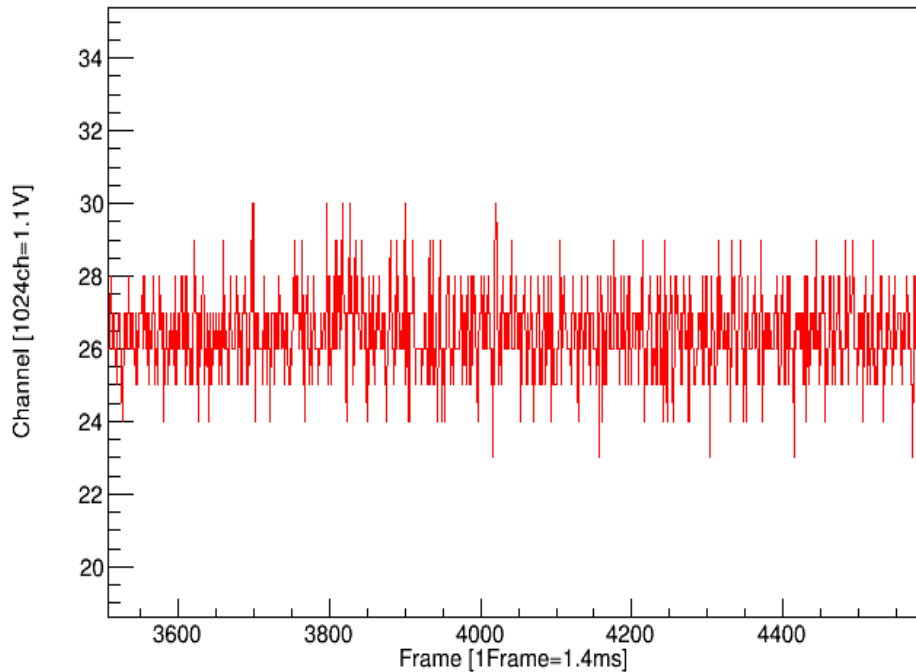


```
makelog5 | Processing 2.1
File Edit Sketch Tools Help
makelog5
import processing.serial.*;
Serial myPort;
String datastr;//mojiretsu gata hennsuu
PrintWriter output;
void setup()
{
  println(Serial.list());//available port list
  myPort = new Serial(this,Serial.list()[1],57600);//set port list
  myPort.clear();
  String datetimestr = nf(year(),2)+"_"+nf(month(),2)+"_"+nf(day(),2);
  output = createWriter(datetimestr+".txt");//open file "log.txt"
}

void draw(){
  if(myPort.available() > 0){//check myPort buffer
    datastr = myPort.readString();//datastr ni jusin mojiretu kakiko
    //println(count);
    //int[] tempdata = int(split(datastr,','));//' kugiri de bunkatu
    //String datetimestr = nf(year(),2)+"/"+nf(month(),2)+"/"+nf(day(),2);
    //String tempstr = String.format("%f,%f",tempdata[0],tempdata[1]);
    print(datastr);
  }
}
```

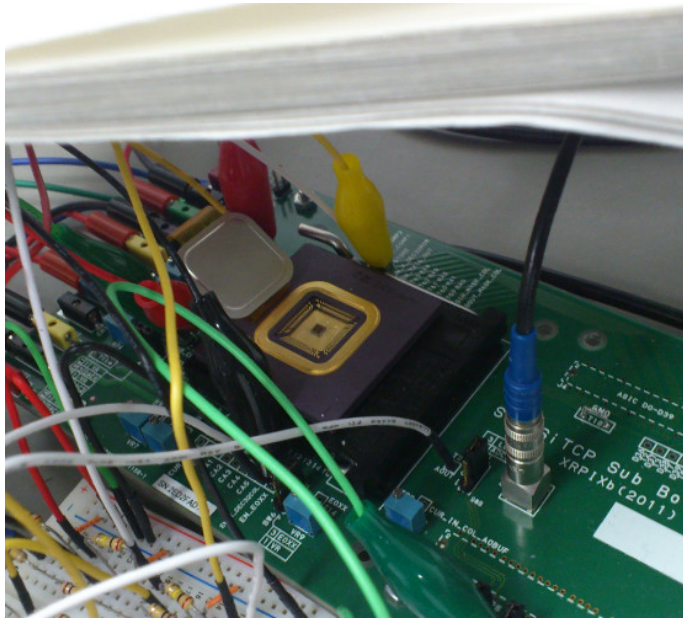
# 1pixel[RA0,CA0]のペDESTアル (遮光状態での出力)

time history

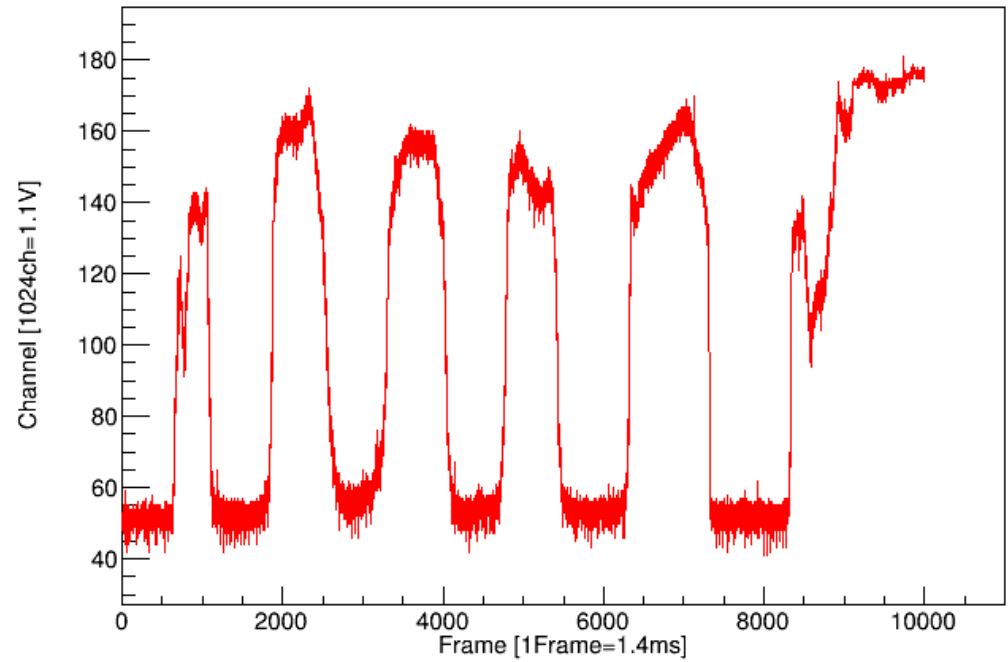




# 蓋を開けて光を遮ってみる



time history



# まとめ

- 1ピクセルに固定しての信号読み出し成功
  - 複数ピクセルを読むための課題として
    - ・マイコンの電圧読み出し時間100 $\mu$ s
- 今回露光時間1msの設定で1フレーム1.4ms。  
1000ピクセル読む場合1フレーム約400msになってしまう。