pid

PID制御を行うクラス

class Pid

void Pid::setGain(double p_gain,double i_gain,double d_gain,double time_constant = 0)

PIDゲインの設定を行う

[パラメータ]
比例制御ゲイン
積分制御ゲイン
微分制御ゲイン
微分前ローパスフィルタの時定数[ms]

[戻り値]
なし

[サンプルコード]

```
#include "stm32f4xx.h"
#include "sken_library/include.h"

Pid pid_control;

int main(void)
{
    sken_system.init();
    pid_control.setGain(1,1,1,20);
    while(1)
    {

    }
}
```

double Pid::control(double target,double now,int control_period)

目標値と現在値からPID制御を行う

[パラメータ]
目標値
現在値
制御周期[ms]

[戻り値]
制御出力

[サンプルコード]

```
#include "stm32f4xx.h"
#include "sken_library/include.h"

Pid pid_control;
double target;
double now;
double out;

void func(void)
{
    out = pid_control.control(target,now,1);
}

int main(void)
{
    sken_system.init();
    pid_control.setGain(1,1,1);
    sken_system.addTimerInterruptFunc(func,0,1);
    while(1)
    {

    }
}
```

double Pid::control(double e,int control_period)

偏差からPID制御を行う

[パラメータ]
偏差
制御周期[ms]

[戻り値]
制御出力

[サンプルコード]

```
#include "stm32f4xx.h"
#include "sken_library/include.h"

Pid pid_control;
double e;
double out;

void func(void)
{
    out = pid_control.control(e,1);
}

int main(void)
{
    sken_system.init();
    pid_control.setGain(1,1,1);
    sken_system.addTimerInterruptFunc(func,0,1);
    while(1)
    {

```

```
    }
}
```

void Pid::reset(void)

積分と微分をリセットする

[パラメータ]
なし

[戻り値]
なし

[サンプルコード]
1秒ごとにPID制御をリセットする

```
#include "stm32f4xx.h"
#include "sken_library/include.h"

Pid pid_control;
double target;
double now;
double out;

void func(void)
{
    out = pid_control.control(target,now,1);
}

int main(void)
{
    sken_system.init();
    pid_control.setGain(1,1,1);
    sken_system.addTimerInterruptFunc(func,0,1);
    while(1)
    {
        if(sken_system.millis() % 1000 == 0){
            pid_control.reset();
        }
    }
}
```