第1章 お手軽Linux時代に高性能とリアルタイムを両立!

IoT時代はラズパイにカチャッ! リアルタイム制御コンピュータ初体験

永原 柊

ラズベリー・パイなど，1GHz動作のプロセッサを搭載した安価な小型コンピュータが増えています．ネットワークに強いLinux OSを搭載し，ネットワーク通信機能を備えているタイプも多く，I/Oも行えるため，モノのインターネット(IoT；Internet of Things)に事足りる場合もあるかもしれません．

しかし実際には，Linuxコンピュータは，処理効率は非常に高いのですが，決められた時間以内に「必ず」処理を済ませることは不得意です．ちょっとした操作・モニタを行うには便利ですが，電子機器の制御・計測を行うには向きません．マイコンを使ったハードウェアのリアルタイム制御・計測が，IoT時代に差をつけるキー・テクノロジとして，ますます重要です．

本稿では，便利な小型Linuxボードであるラズベリー・パイに，制御用マイコン・アダプタをカチャッと追加して，リアルタイム・コンピュータを初体験してみます．(編集部)

IoT時代に知ってないとマズいこと… Linuxはハード制御に使えない

● 実験条件

まずは，ラズベリー・パイ3は制御でどの程度のリアルタイム性能を発揮できるのかを簡単に見てみるために，サーボモータ(ラジコン・サーボとも呼ぶ)を動かしてみます．

超お手軽マイコン基板Arduino Unoと比べてみます．

- マイコン基板Arduino Uno：16MHz動作の8ビットCPU(AVR)
- Linuxコンピュータ ラズベリー・パイ3：1.2GHz動作の4コア32ビットCPU(ARM Cortex-A53プロセッサ)

リスト1 リアルタイム性能をチェックするためのサーボモータ制御プログラム

```
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  while (1) {
    digitalWrite(13, HIGH);
    delayMicroseconds(1000);
    digitalWrite(13, LOW);
    delay(19);
  }
}
```

1msのパルスを出力する

(a) Arduino Uno用

```
#include <wiringPi.h>

#define GPIO18  18

main(int argc, char *argv[])
{
    if (wiringPiSetupGpio() == -1) return 1;
    pinMode(GPIO18, OUTPUT);

    for (;;) {
        // 1ミリ秒の幅のパルスを出力
        digitalWrite(GPIO18, 1);
        delayMicroseconds(1000);
        digitalWrite(GPIO18, 0);

        //残り19ミリ秒待つ
        delayMicroseconds(19000);
    }
}
```

(b) ラズベリー・パイ3用(wiringPiライブラリのdelayMicrosecondsHard関数使用)

```
#include <wiringPi.h>

#define GPIO18  18

main(int argc, char *argv[])
{
    if (wiringPiSetupGpio() == -1) return 1;
    pinMode(GPIO18, OUTPUT);

    for (;;) {
        // 1ミリ秒の幅のパルスを出力
        digitalWrite(GPIO18, 1);
        delayMicrosecondsHard(1000);
        digitalWrite(GPIO18, 0);

        //残り19ミリ秒待つ
        delayMicrosecondsHard(19000);
    }
}
```

(c) ラズベリー・パイ3用その2(delayMicrosecondsHard関数使用)