

# シェルスクリプトを用いたUNIX哲学に基づくリアルタイム制御 **Real-time Control Using Shell Script Based on The UNIX Philosophy**

2021年6月2日

USP研究所 柳戸新一\* 松浦智之 鈴木裕信

金沢大学 大野浩之

ソフトウェア・シンポジウム2021

---

# 論文の正誤表

---



P6. 左列最下行

誤 : 「 $\Delta t = 0.01 \text{ ms}$ 」

正 : 「 $\Delta t = 10 \text{ ms}$ 」

- **IoT機器の高機能化に伴い、組込Linuxの使用へ**
  - ネットワークへの接続
  - 周辺機器との通信
- **長期に渡って利用され続ける組み込み機器の脆弱性が問題に**
  - ソフトウェアの持続性・移植性が重要視
- **OSの更新に左右されにくいシェルスクリプト（特にPOSIXに準拠したもの）をベースにした開発手法が注目されている**

# 本研究の目的

---



usp lab.

## 提案手法

持続性が高いとされるシェルスクリプトを用いた、リアルタイム制御手法の提案

## 検証方法

提案手法を用いた、倒立振子の安定化

## リアルタイム制御における互換性向上

リアルタイムカーネル

(RTLinux/ART-Linux)

休眠システムコール  
による周期的な処理

[熊谷ら 2004]

## ソフトウェアの持続性向上

POSIX中心主義  
プログラミング

シェルスクリプト による  
POSIX規格 に準拠した  
開発手法  
[松浦ら 2017]

ものグラミング

松浦らの手法を  
ものづくり に適用

[大野ら 2017]

大野らの手法を  
(リアルタイム) 制御  
に適用

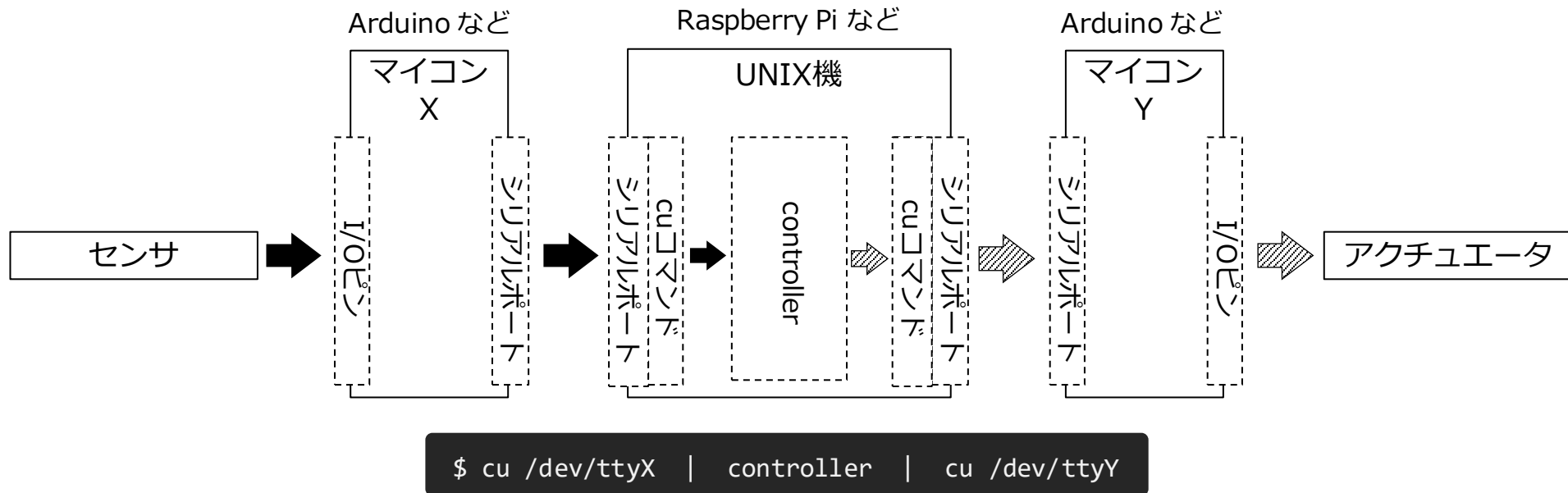
(本研究)

## ものグラミング2 - 互換性・持続性に配慮したマイコン間の通信手法

データを変換用に、UNIX機を用いる

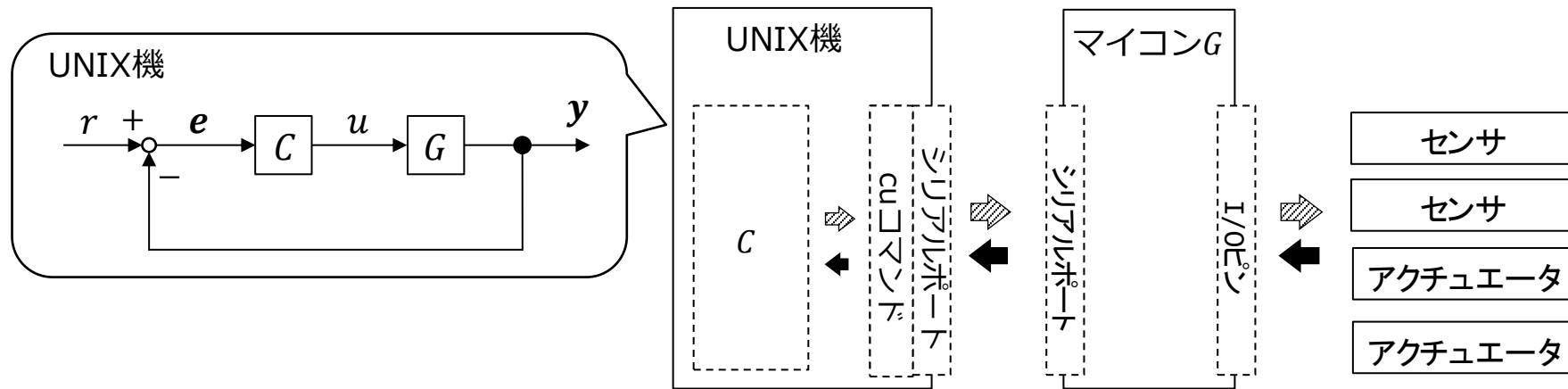
⇒ 制御データの流れ

➡ センサ値の流れ



# 本研究で扱うフィードバックシステム

- 「ものグラミング2」に則り、役割分担
  - マイコン：センサ・アクチュエータ管理
  - UNIX機：制御量計算
- UNIX機ではLinuxを用いる、カーネルをいじらない（持続性向上）



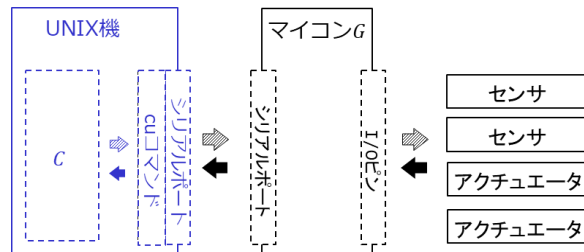
シェルスクリプト（名前付きパイプ）を用いたフィードバック制御手法

# 提案手法

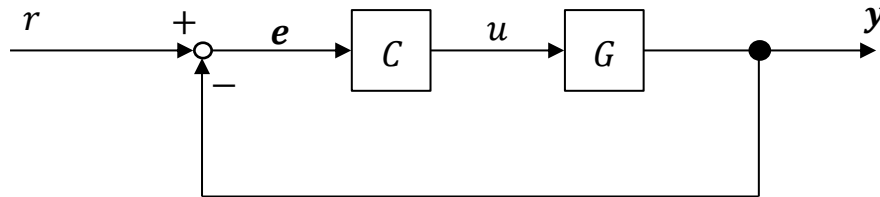
## シェルスクリプトを用いたフィードバック制御



usp lab.



UNIX機



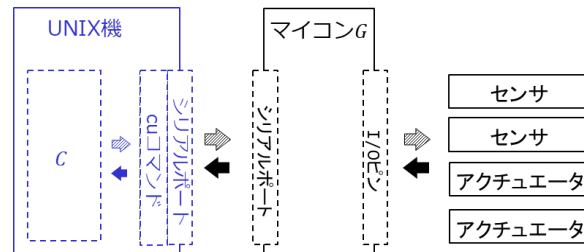
\$ # シェルスクリプトで表したいフィードバック制御 ↑



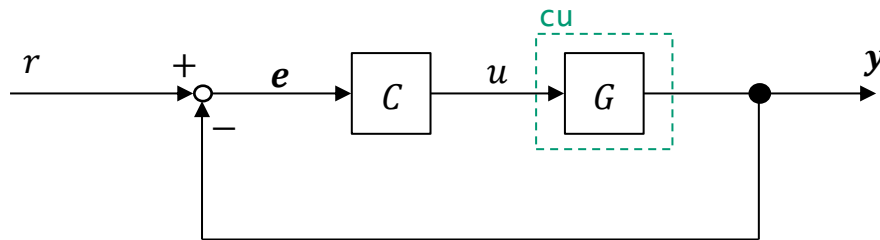
# 提案手法



## シェルスクリプトを用いたフィードバック制御



UNIX機



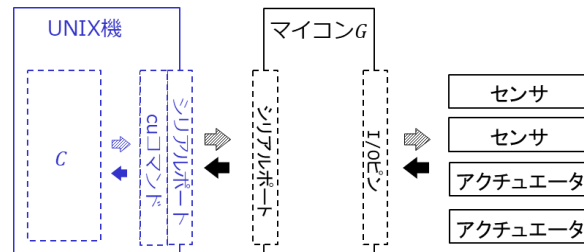
\$ # cu コマンドを用いて G に入出力

\$

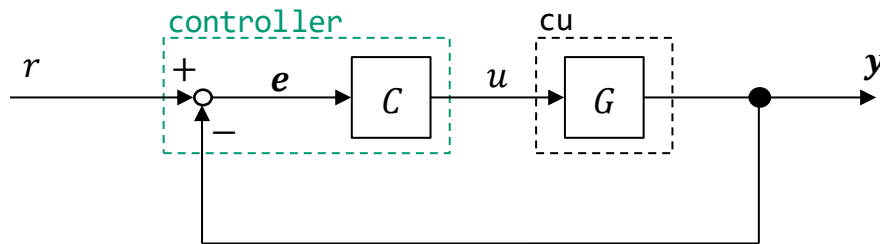
\*\*\* | cu /dev/G | \*\*\*

# 提案手法

## シェルスクリプトを用いたフィードバック制御



UNIX機



```
$ # 制御器 controller の出力を cu の入力へ
```

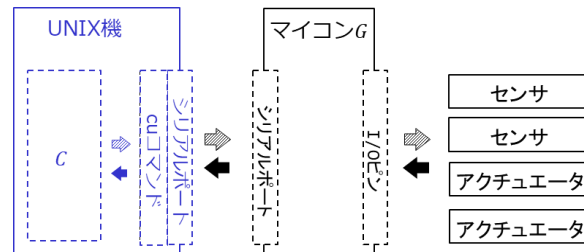
```
$          *** | controller          | cu /dev/G |          ***
```

# 提案手法

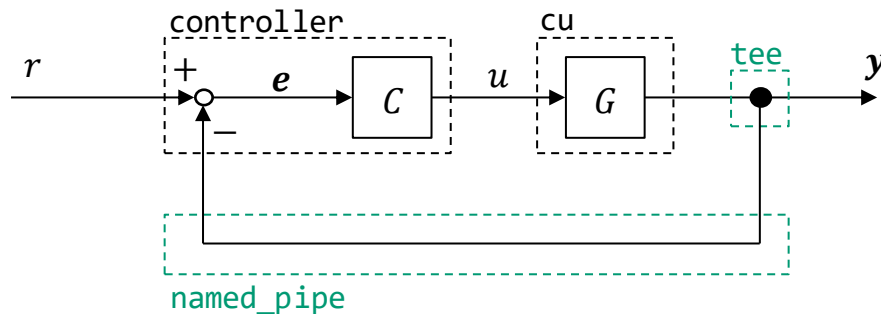
## シェルスクリプトを用いたフィードバック制御



usp lab.



UNIX機



\$ # 分岐点は tee コマンド、フィードバックは名前付きパイプ

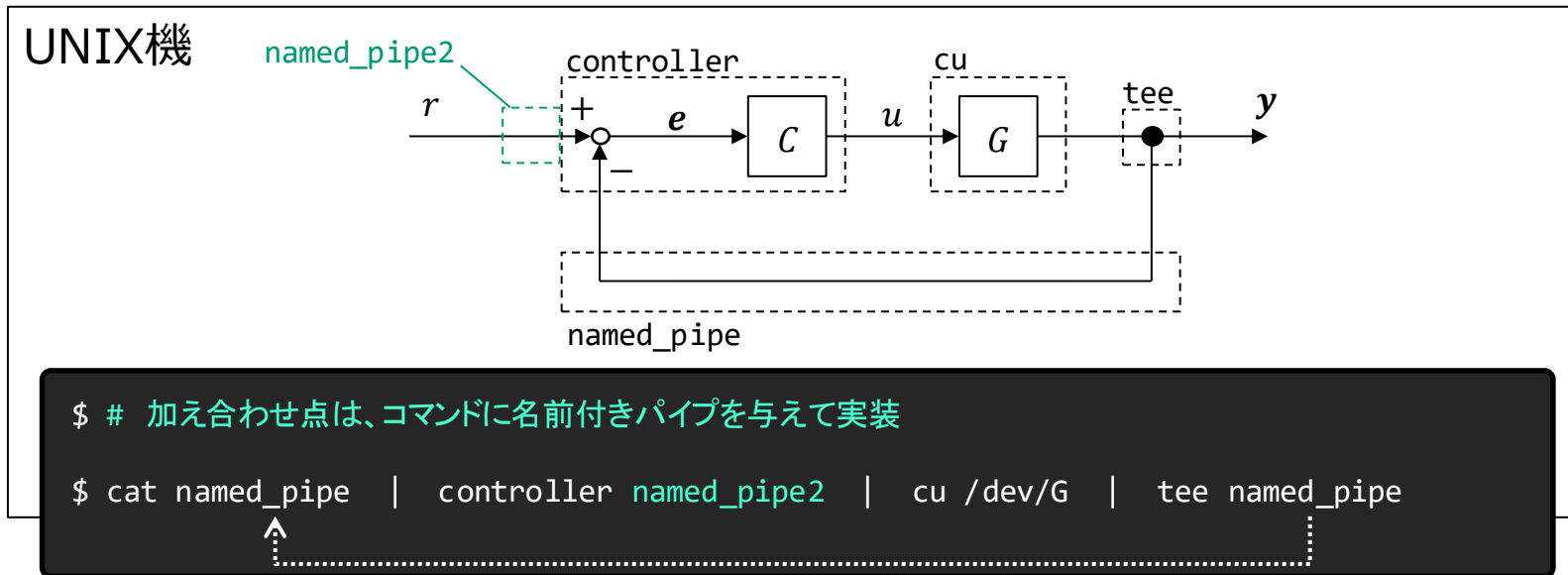
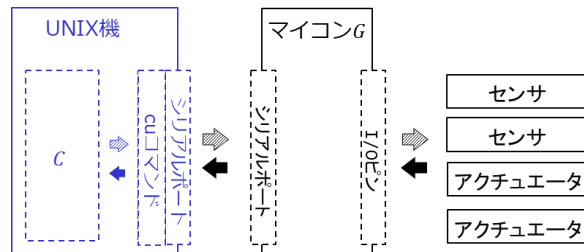
```
$ cat named_pipe | controller | cu /dev/G | tee named_pipe
```

# 提案手法

## シェルスクリプトを用いたフィードバック制御



usp lab.

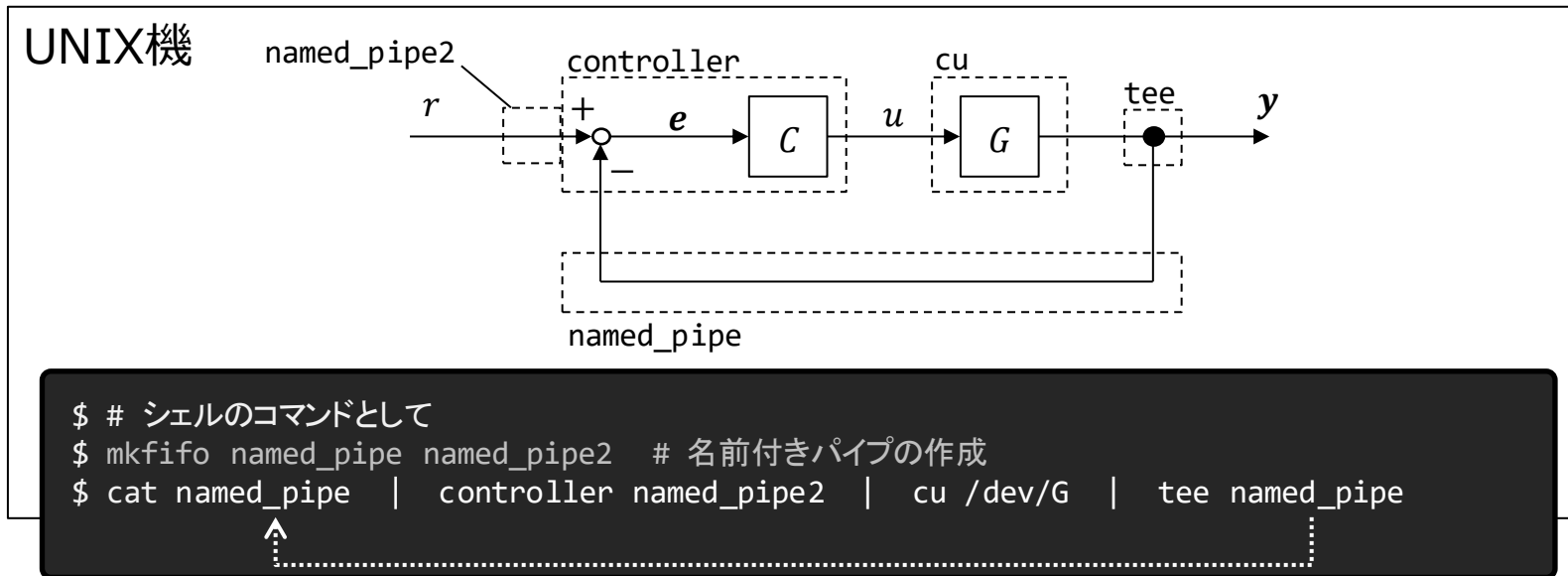
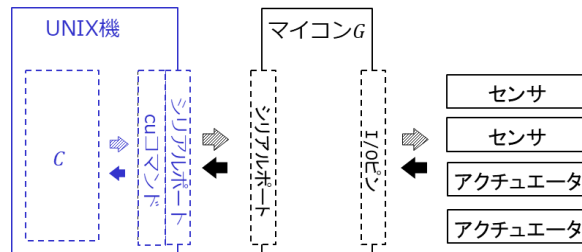


# 提案手法

## シェルスクリプトを用いたフィードバック制御



usp lab.



# UNIX哲学 [Gancarz 1995] との関係（9の定理）

---




usp lab.

## 9つの定理のうち、満たしているもの

- 各機能に特化したプログラム（コマンド）をつなぎ合わせて実行（定理1、定理2）
- 効率<移植性（定理4）
- 数値データはASCIIで出力（定理5）
- シェルスクリプトの利用（定理7）
- プログラムをフィルタとして設計（定理9）

```
$ # シェルのコマンドとして
$ mkfifo named_pipe named_pipe2 # 名前付きパイプの作成
$ cat named_pipe | controller named_pipe2 | cu /dev/G | tee named_pipe
```



## シェルスクリプト

持続性・互換性の向上（本研究の主目的）

## パイプライン

ブロック線図の自然な実装

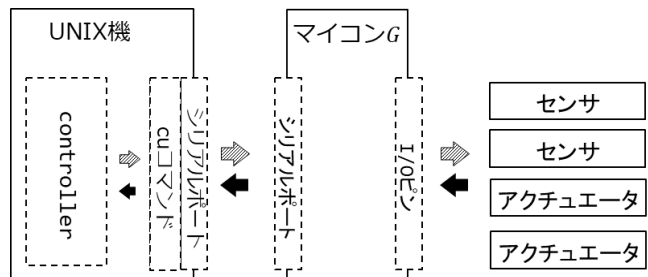
## コマンド

入れ替えによって容易に制御アルゴリズムを変更可

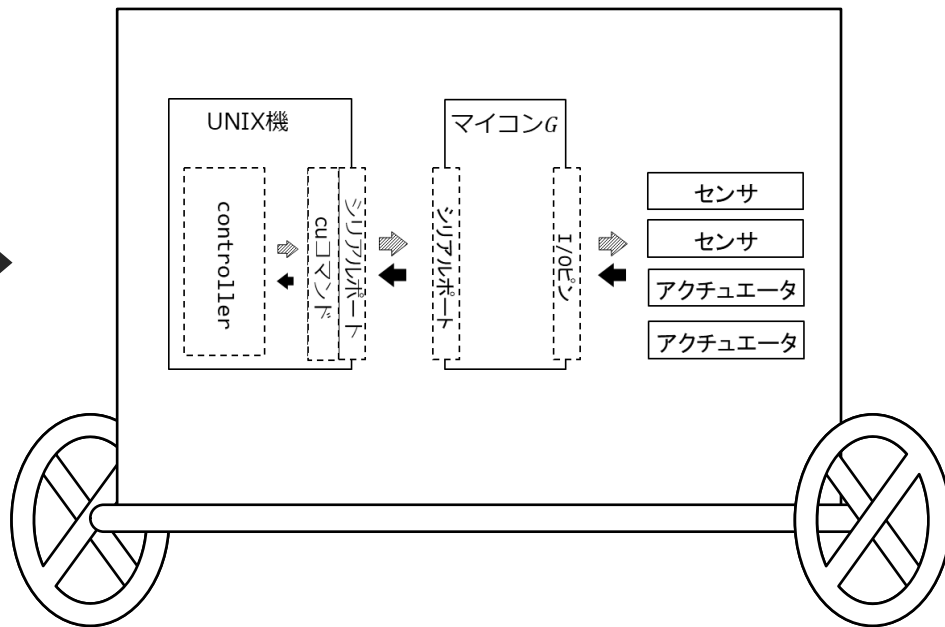
# 検証のために倒立振子の安定化

特に、同軸2輪車型と呼ばれるもの

## 提案手法



## 有効性の検証方法





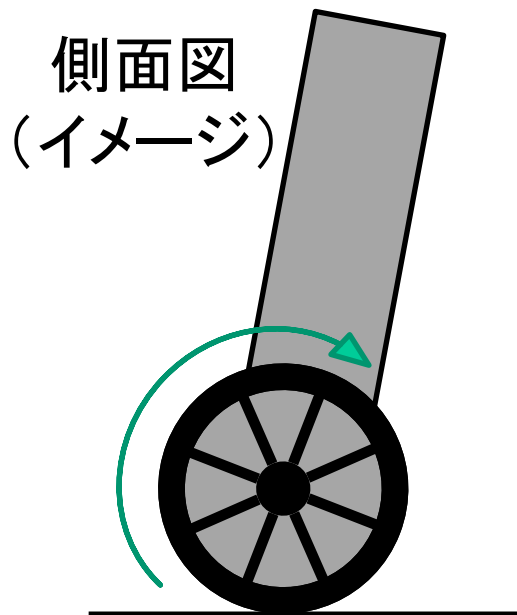
# 倒立振り子とは



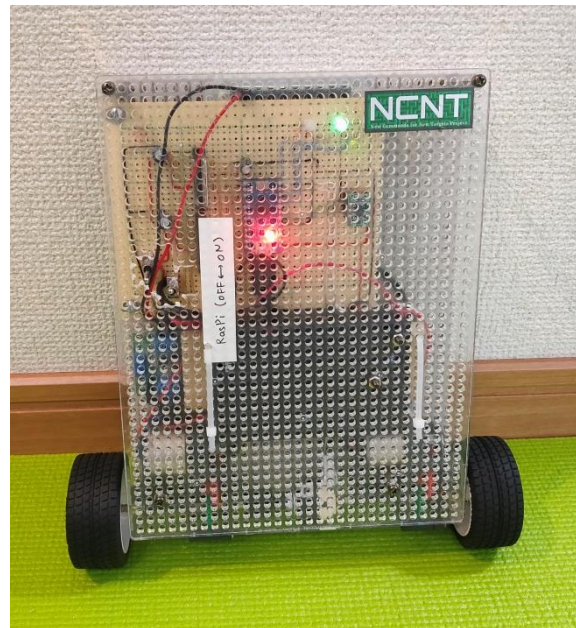
usp lab.

## 自立する二輪車

制御しないと転倒するが、うまく制御することにより倒立させられる



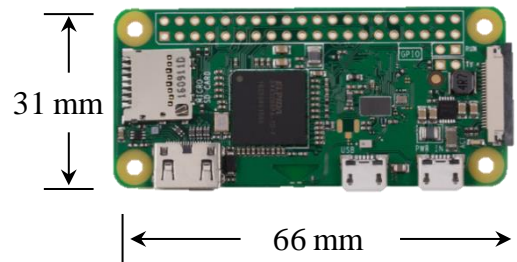
## 正面図



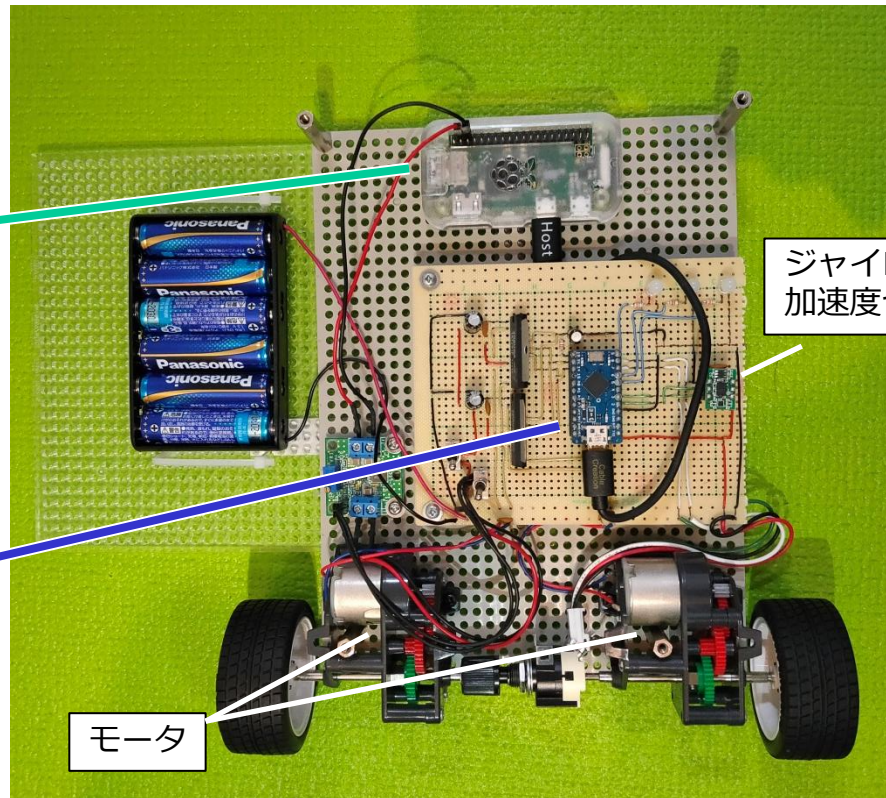
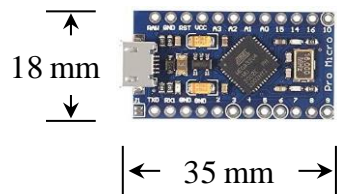
\*「倒立振り子ロボット ver2を作る (6) -位置制御機能の実装」  
を参考に作成

# 実際に組み立てたものがこれ

Raspberry Pi Zero (UNIX機)



Arduino Pro Micro (マイコン)



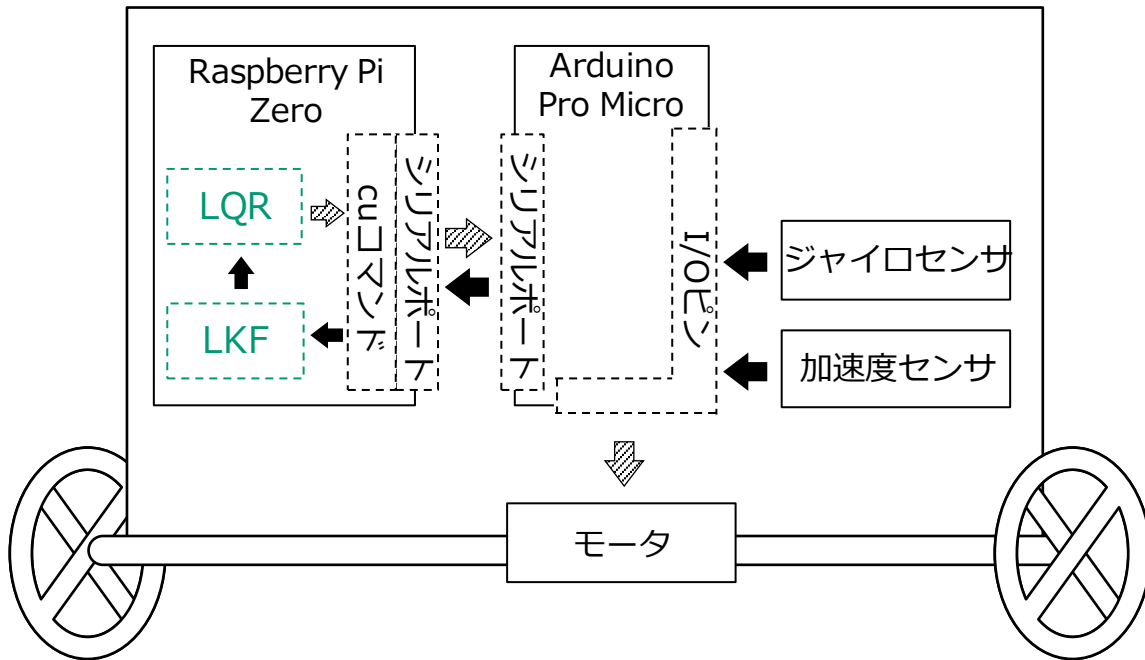
デモ動画



<https://www.youtube.com/watch?v=Ixxh-YFtMypY>

# 安定化のための制御コマンド

controller の代わりに2つのコマンドを作成・利用



**LQR** (線形二次レギュレータ)

➤ 制御量の計算

**LKF** (線形カルマンフィルタ)

➤ ノイズの除去

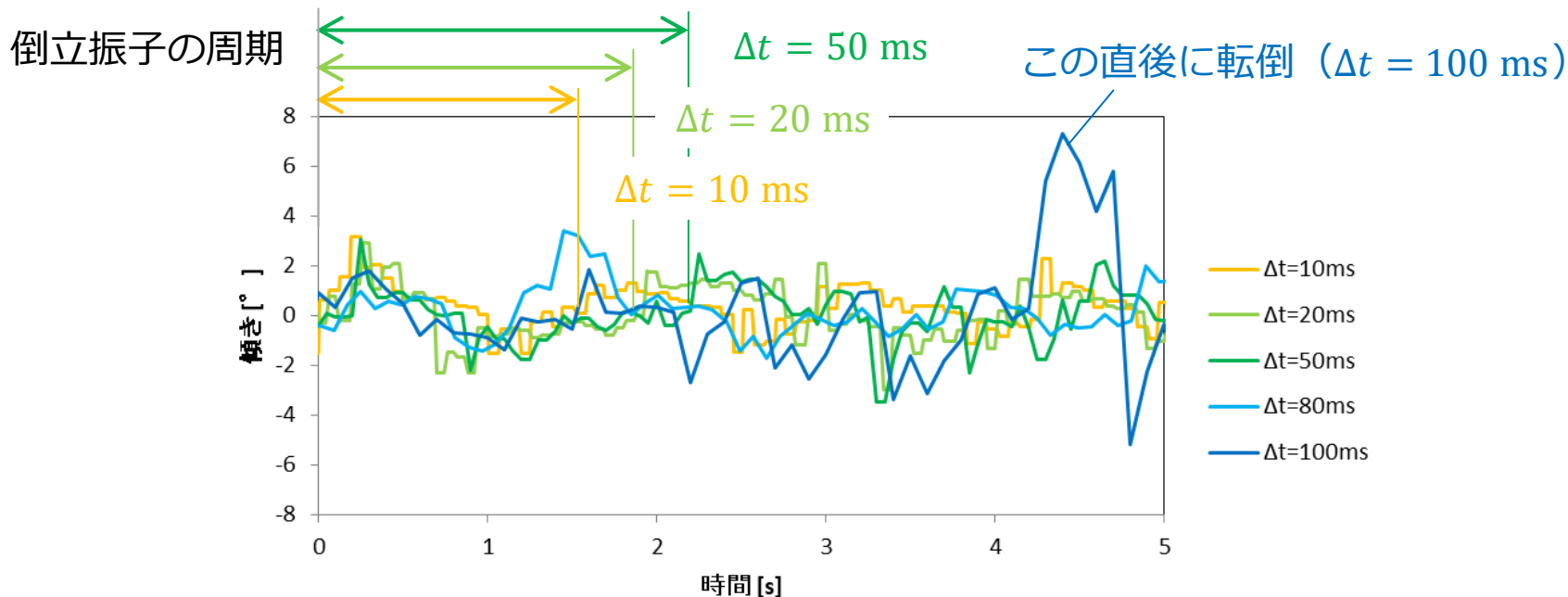
LKFは計算のためにサンプリング間隔  $\Delta t$  が必要

- $\Delta t = 10 \text{ ms} \sim 100 \text{ ms}$  で動作検証

# サンプリング間隔 $\Delta t$ 変更時の倒立振子の挙動



usp lab.



- $\Delta t$  が短いとより高速な反復運動をする

- シェルスクリプトを用いたリアルタイム制御手法を提案した
  - ブロック線図のデータの流れを、シェルスクリプトのストリームで実装
- 提案手法で同軸2輪車型倒立振子を安定化させられた
- 高い持続性・互換性を制御ソフトウェアにもたらせる可能性を示した

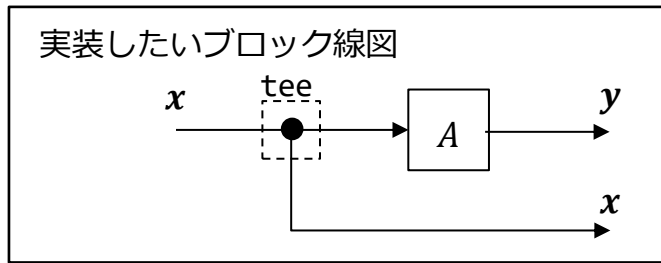
- 外部からの指示による倒立振子の前後移動
- 本手法が適用可能な条件
  - サンプルング間隔  $\Delta t$  がより小さい場合の倒立振子の挙動
  - 倒立振子の物理的特性と提案手法との関連
- 倒立振子の外部からの操作・平面上の移動





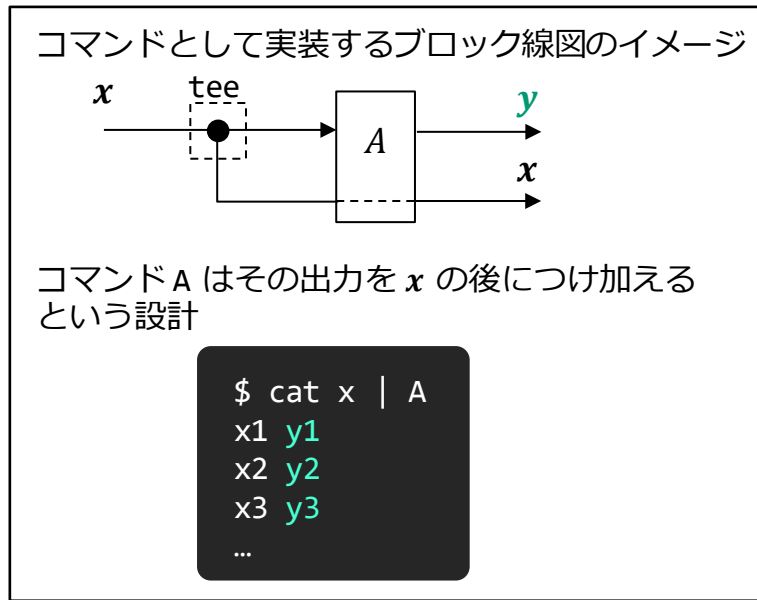
## 複雑なブロック線図に対して、機械的なアルゴリズムとして提案手法を適用する方法

分岐点後の並列処理を直列に直す。



$x$  のデータ

```
$ cat x
x1
x2
x3
...
```



### 正常処理においてはシンプルなパイプラインとして構成できても、 エラー時の処理はどのようなのか

いくつかエラー処理の方法があります。

1. エラー処理のモジュールをブロック線図に組み込む
2. エラーであることがわかるように出力し、その後ある時刻の出力をすべて破棄

```
$ cat np | cmd1 | drop-if-error | cmd2 | ... | tee np
```

3. 標準エラーに出力された結果を、パイプラインの外で処理

```
$ (cat np | cmd1 2>&3 | cmd2 2>&4 | ... | tee np) 3>cmd1.log 4>cmd2.log ...
```

同様の倒立振子を作る場合、値段はいくらくらいか

2万円程度

合計	約20000円
Raspberry Pi Zero	1320
Arduino Pro Micro	1980
電源、モータ、センサ他	約16000

**提案手法が容易に使えるようになるためのツール群を作成する場合、その基礎となる制御用コマンドにはどのようなものが考えられるか**

現場ではPID制御が最もよく使われるので、PID制御のコマンドが基本となるでしょう。

**提案手法はブロック線図に比べるとデータの流が少々見えにくくなって  
しまうのではないか**

ブロック線図から作成しているので、データの流は多少わかりにくくはなります。  
。制御用のコードを書いても基にしたブロック線図は残しておくのがいいです。  
。

**提案手法はリアルタイム制御ということだが、入力が出てから出力されるまでの保証をすることができるのか**

現状は、持続性・移植性向上とのトレードオフになっており、保証することは難しいと思います。なお、本研究では移植性・持続性向上のために標準のLinuxカーネル（カーネルバージョン4.19.66のデフォルトスケジューリングアルゴリズムはCFS）を用いることを前提としています。加えて、コマンドでも優先度の変更等はいりません。