

渡部研究室

Programming Systems Group

主な研究テーマ

- ・ プログラミング言語
- ・ 形式手法
- ・ 組み込みシステムとCPS

最近の研究

- ・ 組み込みシステムとCPSのための関数リアクティブプログラミング言語
- ・ 逆計算・形式手法

教員

渡部卓雄（教授）

プログラミング言語の設計と実装, メタプログラミングとリフレクション, アクターモデルと並行オブジェクト, 関数プログラミング

森口草介（助教）

ソフトウェアおよび各種システムの形式的記述と検証, 定理証明支援系, プログラミング言語の意味論, 抽象議論フレームワーク

学生(2024年1月現在)

博士2, 修士4, 学士4

<https://www.psg.c.titech.ac.jp>



プログラミング言語の研究

- **理論**：プログラミング言語の意味論, 型システム等
 - プログラムやプログラミング言語の性質の理解
 - プログラムの検証
- **言語設計**：新しい言語や言語機構の提案
 - プログラムの開発効率の向上 (評価は難しい)
 - プログラムがある特定の性質をもつことを保証 (例：型安全性, メモリ安全性)
- **実装技術**
 - 実行速度の向上, 省メモリ化, 低消費電力化

組込みシステムとCPS向けの 関数リアクティブプログラミング(FRP)言語

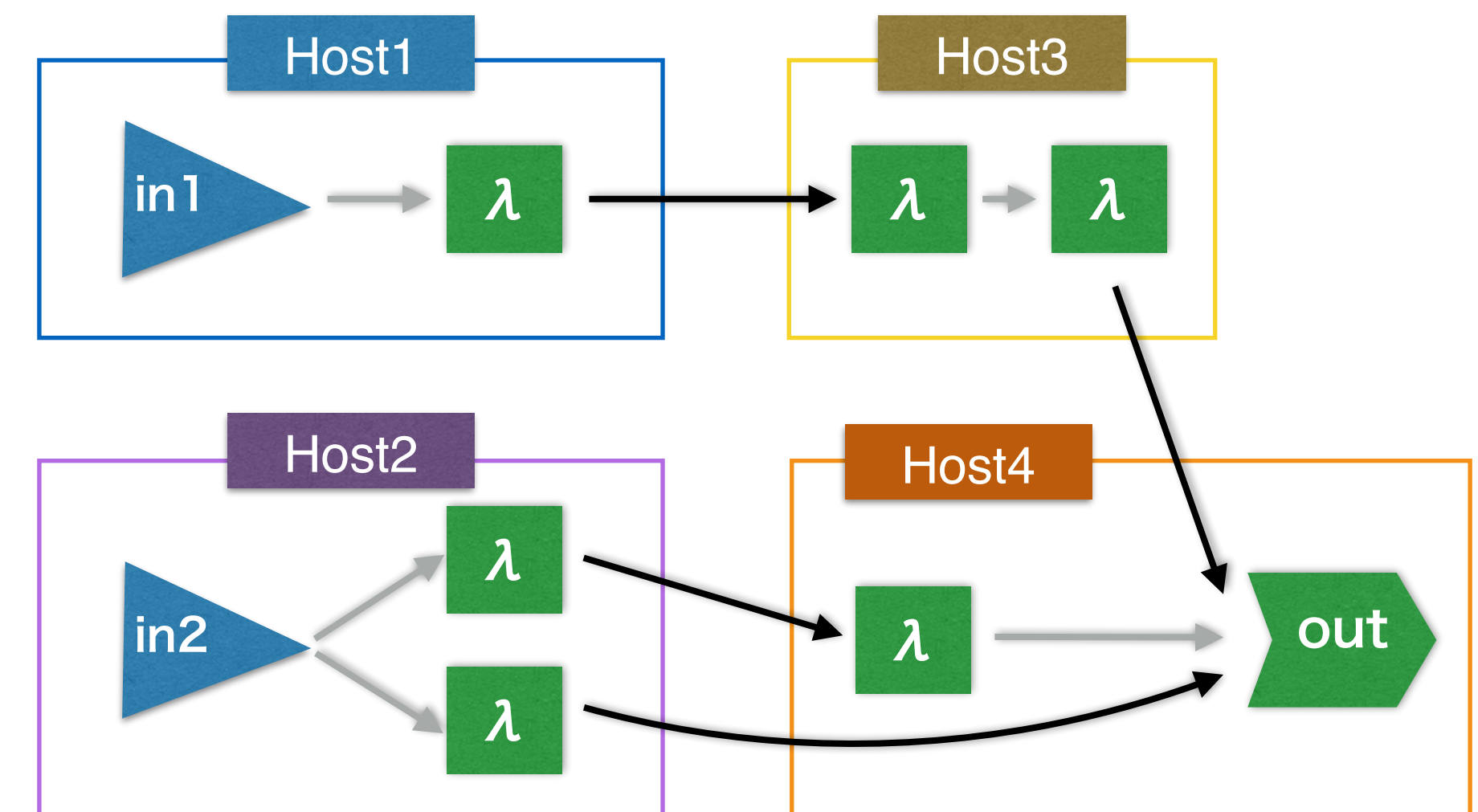
- **目的**：組込みシステム・CPSの開発効率・安全性の向上
 - CPS: サイバーフィジカルシステム
- **チャレンジ**
 - 言語設計：表現力と静的特性のバランス，モジュール性，非同期処理
 - 実装：低速・低機能CPU，少メモリ，省電力化，分散化，高信頼化

• 現在のプロジェクト

- 実時間処理，低消費電力化，分散化
- リソース制約を保證する型システム

• 今後の課題

- 形式検証



従来の言語を使った 組み込みシステムの記述例

コールバック

例題：温度・湿度センサの値から不快指数を求め、その値が75以上の時にファンを回す。

ポーリング

```
Sensor tmp(TEMP_SENSOR);
Sensor hmd(HUMID_SENSOR);
GPIO fan(FAN_SWITCH);

while (true) {
    float t = tmp.read();
    float h = hmd.read();
    float di = 0.81 * t + 0.01 * h
        * (0.99 * t - 14.3) + 46.3;
    fan.write(di >= 75.0);
}
```

入力→計算→出力の繰返し。反応速度は遅い入力に依存する。複雑な動作の記述には向かない。

```
Sensor tmp(TEMP_SENSOR);
Sensor hmd(HUMID_SENSOR);
GPIO fan(FAN_SWITCH);
float t, h;
bool fan_state = false;

tmp.onChangeed(lambda(float v) {
    t = v; changeState(); });

hmd.onChangeed(lambda(float v) {
    h = v; changeState(); });

synchronized void changeState() {
    float di = 0.81 * t + 0.01 * h
        * (0.99 * t - 14.3) + 46.3;
    bool s = di >= 75.0;
    if (fan_state != s) fan.write(s);
    fan_state = s;
}
```

入力が変わったときに呼ばれる関数を記述する。動作が複雑になるとプログラムがかなり複雑になる（callback hell）。

関数リアクティブプログラミング(FRP)

- 関数プログラミングの考え方にもとづいて、リアクティブシステムの記述を支援するプログラミングパラダイム
 - リアクティブシステム：外界から連続的・非連続的に与えられる入力に反応し続けるシステムのこと。入力のタイミングや順番は予測できない。
 - 例：GUI, 組込みシステム, サイバーフィジカルシステム(CPS)
 - 対義語：変換的(transformational)システム。有限な入力に対して答を出力して停止する（古典的な計算モデルにもとづく）システム。
 - FRP言語の例：Fran [Elliot '97], Yampa [Hudak '03], Elm [Czaplicki, '12], etc.
- ポイント：時間とともに変化する値を時変値(Time-Varying Value)として表現し、その上の計算としてリアクティブな動作を記述する。

FRP言語による記述

本研究室で開発した組込みシステム向けFRP言語 Emfrpによる記述.

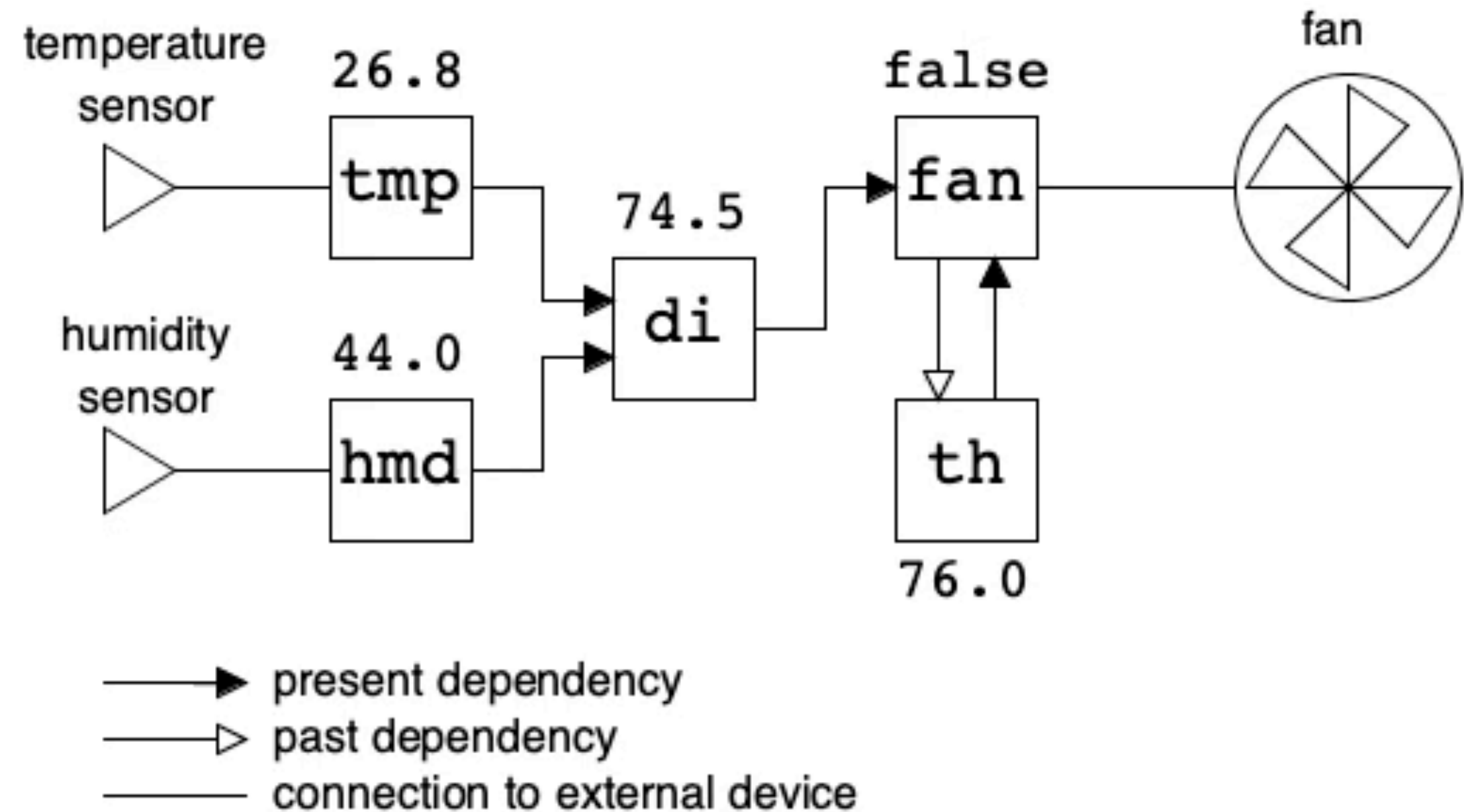
```
module FanController
in  tmp : Float # 温度センサ
    hmd : Float # 湿度センサ
out fan : Bool # ファンのスイッチ
use Std

node di = 0.81 * tmp + 0.01 * hmd
        * (0.99 * tmp - 14.3) + 46.3;

node init[False] fan = di >= th

node th = 75.0 +
    if fan@last then -1.0 else 1.0
```

プログラム中のtmp, hmd, di, fan, th は時変値であり、時間と共に変化する。このプログラムでは、ファンのモータを保護するためのヒステリシス制御も行っている。

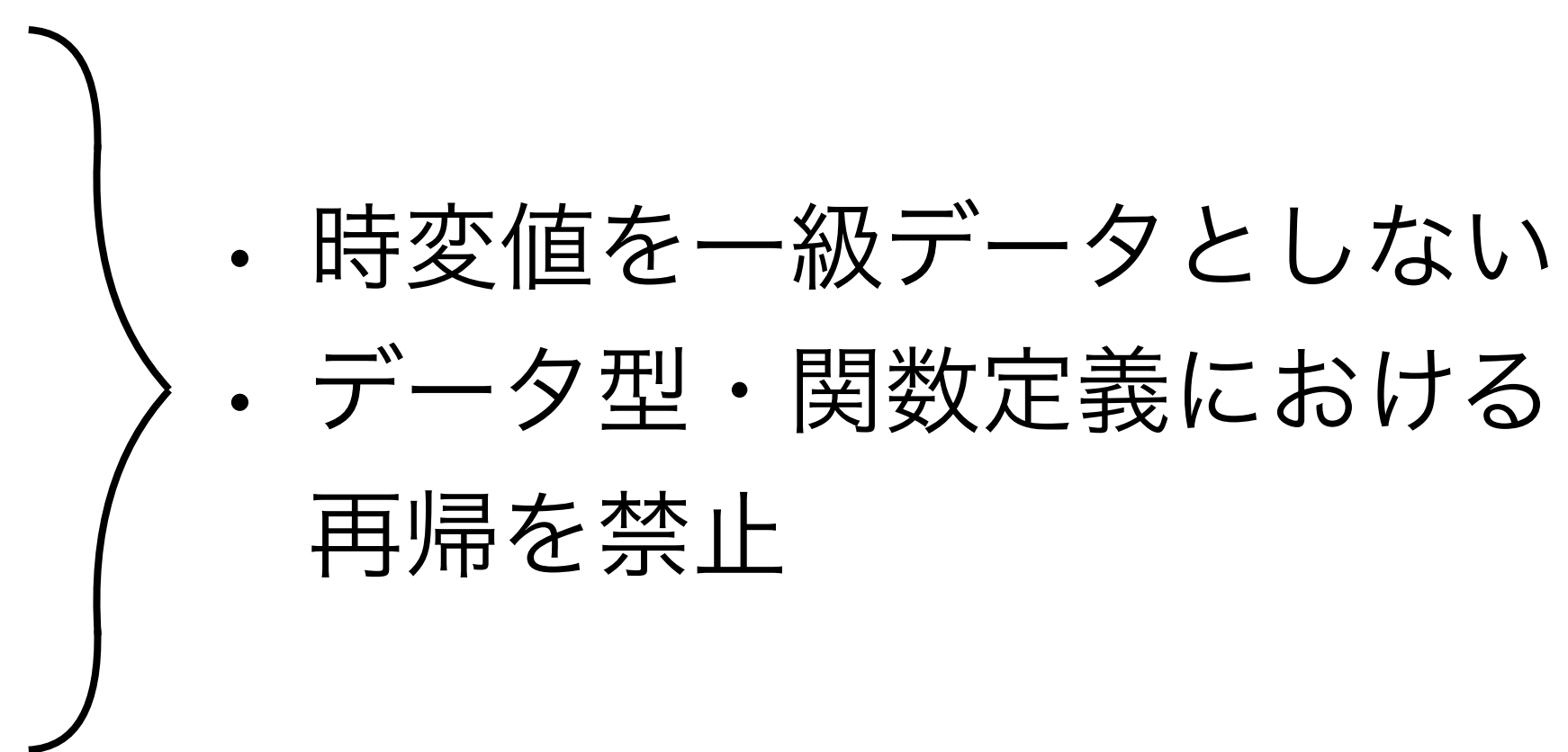


Emfrpのプログラムは、時変値を節点、それらの依存関係を有向辺とする有向グラフとして図式化できる。fanからthへの辺は、thの定義におけるfanの直前値の参照（プログラム中のfan@last）を表している。この仕組みにより、Emfrpは副作用を持たない関数型言語にもかかわらず状態を持つ動作を記述できる。

Emfrp処理系:

github.com/psg-titech/emfrp

Emfrpの特徴

- 純粋な関数リアクティブプログラミング言語
 - 静的型, 型推論, 代数的データ型
 - コールバックやリフティングが不要 (時変値と普通の値を混在できて直感的に書ける)
 - マイクロコントローラなどの小規模組込みシステム向けに設計
 - ソースコードは標準的なCにコンパイルされる
 - プラットフォーム依存の部分とリンクすることで実行可能コードを生成
 - 実行時のメモリサイズが静的に決定される
 - 実行中にメモリ不足になることはない
 - 時変値の更新に用いる式の計算の停止性を保証
 - 実行中にハングして無反応になることはない
- 
- 時変値を一級データとしない
 - データ型・関数定義における再帰を禁止

倒立振り子ロボットの記述例

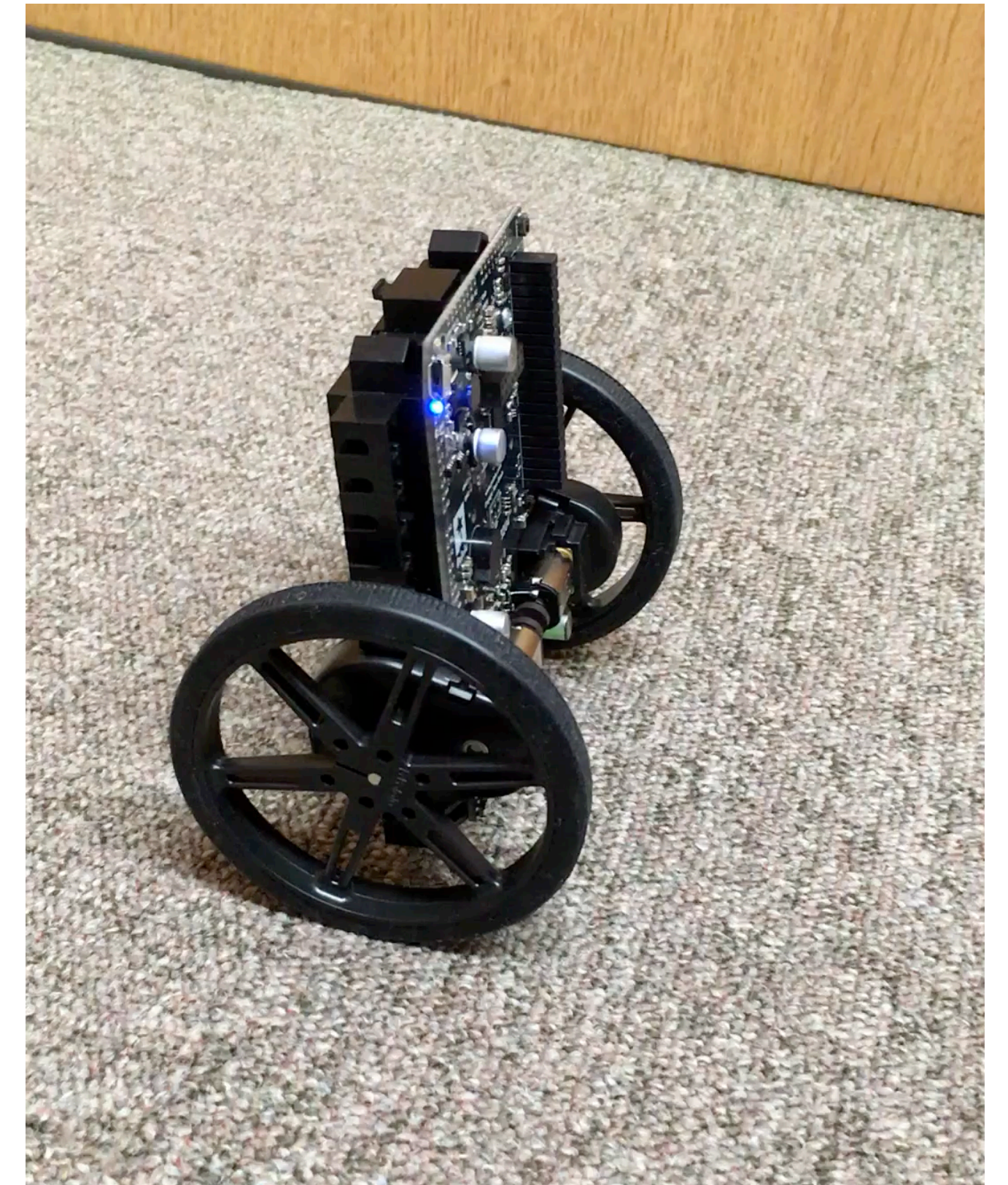
```
module Balancer
in  gyroY  : Int,  # gyroscope (y-axis)
    accX   : Int,  # accelerometer (x-axis)
    encL   : Int,  # left motor encoder
    encR   : Int   # right motor encoder
out motorL : Int,  # left motor
    motorR : Int   # right motor
use Std
...           # definitions of constants

node init[0] angle =
    (angle@last + gyroY * update_time_ms) * 99 / 100

node speedL = encL - encL@last
node speedR = encR - encR@last
node init[0] distL = distL@last + speedL
node init[0] distR = distR@last + speedR

node risingAngleOff = gyroY * angle_rate_ratio + angle
node init[0] motor =
    motorSpeed(motor@last +
        (angle_resp * risingAngleOff +
         dist_resp * (distL + distR ) +
         speed_resp * (speedL + speedR)) / 100 / gear_ratio)

node diffSpeed = (distL - distR) * dist_diff_resp / 100
node motorL = if accX > 0 then motor + diffSpeed else 0
node motorR = if accX > 0 then motor - diffSpeed else 0
```



Pololu Balboa 32U4
(ATmega 32U4, 32KB
Flash, 2.5KB RAM)

github.com/psg-titech/emfrp_samples

なぜ新しい言語なのか？

- 構文や型による（強力な）制約が欲しい
 - 「この言語で書いたプログラムではこういう性質が常に成立する」と言いたい
 - Emfrpの場合：メモリ制約, 更新処理の停止性, 実時間性, etc.
- 「C++ / Rust / Python / Haskell, etc の拡張ではダメなの？」
 - 時変値（あるいはシグナル関数）を既存言語のオブジェクトや関数として表現する必要
 - $x = y + z$ を $x = \text{app2 lift2}(+) (y, z)$ のように書きたくない
 - 新しいアイデアが既存の言語上で必ずしも素直に表現できるとは限らない
 - 例) BCT (データの大きさを静的に規定する型システム)

FRP言語に関する最近の研究

- Emfrp/S : 周期的タスクの効率的な実行機構 [[ICSCA'24](#)]
- TEFRP : タイミング記述方式の提案 [[WCTP'23](#)]
- 低電力コプロセッサの活用 [[SWEST25](#)][[APRIS'23](#)]
- EvEmfrp : 周期的・非周期的タスクの記述/実行方式 [[REBLS'23](#)]
- Multiparty FRP : 分散型FRPのための同期手法の提案 [[WSSE'23](#)]
- Emfrp-REPL : マイクロコントローラ上で動作するインタプリタ [[MoreVMs'23](#)]
- 非同期タスク機構の導入 [[REBLS'22](#)]
- Emfrp^{BCT} : 使用メモリ量を規定できる再帰的データ型 [[JIP'21](#)]
- XFRP-G : GPGPU上でのFRPの実行方式 [[ICSCA'21](#)]
- XStorm : 状態依存動作のための抽象化機構 [[TPRO'20](#)]

- 実時間性の保証
- 静的性質の保証
- 低電力化
- 分散化・並列化
- モジュール性向上

逆計算

- 出力から入力を計算すること
 - 可逆計算：出力から入力が一意に定まるような計算
 - 量子計算, 可逆回路等
 - プログラミング言語関連
 - 可逆プログラミング言語 (例: Janus)
 - 互いに逆になることが保証された2つのコードの記述手法
 - 応用: 圧縮/展開, エンコード/デコード, デバッグ
- 同期的データフロープログラミングにおける逆計算
 - リアクティブシステムのための逆計算方式の提案 [コンピュータソフトウェア'24]
 - Agdaを使った検証

```
procedure fib(int x1, int x2, int n)
  if n = 0 then
    x1 += 1
    x2 += 1
  else
    n -= 1
    call fib(x1, x2, n)
    x1 += x2
    x1 <=> x2
  fi x1 = x2

procedure main()
  int x1
  int x2
  int n
  n += 4
  call fib(x1, x2, n)
```

Janusによるプログラムの例

プログラミング言語から形式手法へ

- 形式手法(formal method)
 - 数学（論理学）に基づき信頼性・頑健性の高いソフトウェアを構築するための各種手法
 - 形式仕様記述, 検証, 解析など
- 渡部研究室における研究の方向
 - プログラミング言語設計やソフトウェア工学分野で発展してきた抽象化・モジュール化手法を積極的に取り入れることで, 形式手法をより身近で実用的なものにしたい.
- 渡部研における主なプロジェクト
 - Actario: アクターモデルのCoqによる形式化 [[AGERE'15](#)]
 - 検証済みプログラムの進化と発展 [[SAC'23](#)]
 - Moxa: AOPにもとづくスケーラブルな仕様記述言語(2005-2007)

最近の卒業論文と修士論文

卒業論文（学士特定課題研究報告書）

2023

- REPLをサポートする小規模組込み機器向けFRP言語処理系の実装と評価（鈴木豪）
- 組込みシステム向けFRP言語における離散イベント機構の実装とその評価（十河健人）

2022

- 組込みシステム向け関数リアクティブプログラムに対するテストとデバッグ手法（内野駿亮）
- サイズ情報を伴った再帰データ型を扱う小規模組込みシステム向けFRP言語へのパラメータ多相の導入（白井瑞貴）
- 組込みシステム向けFRP言語における入出力の動的な切替機構（瀧本哲史）

2021

- 関数リアクティブプログラミング言語のための時間制約付きイベントの記述方式（堀紗知子）
- 組込みシステム向けFRP言語におけるモデル検査を用いた状態依存動作の検証（内藤博）

修士論文

2023

- Glitch-Free Distributed Functional Reactive Programming in Resource-Constrained Reactive Distributed Systems（居桂園）
- 関数リアクティブプログラミングにおける時間制約を持つイベント列の形式的表現（堀紗知子）
- 状態依存動作の記述を支援するFRP言語のための検証用モデル生成手法（内藤博）

2022

- 時変値更新周期の揺らぎを考慮した関数リアクティブプログラミング言語の安全性検査（白鳥佑弥）
- 組込みシステム向けFRP言語における離散イベントの時間的特徴付けと実装方式（辻悠太）

2021

- 無線センサーアクターネットワーク向けマクロプログラミング言語（後藤司）
- 小規模組込みシステム向け関数リアクティブプログラミング言語のための再帰データ型（横山陽彦）

B4の1年

- 全体ゼミ（全員参加）週2回（曜日と時間は毎学期更新）
 - 4月～9月はM1以上が発表、それ以降はB4も発表
 - 前期は週2回のうち1回は輪講
- 輪講：4月～9月
 - B4は全員参加、M1以上は任意（現状はほぼ全員参加）
- プログラミングプロジェクト：4月～5月
 - プログラミング言語研究のためのトレーニングとして、簡単なプログラミング言語処理系（インタプリタやランタイムシステム）の作成を行う
- 学士特定課題研究
 - プログラミングプロジェクトが終わったあたりから

どんな人に向いているか

- プログラミング言語に興味のある人
 - 言語設計と実装技術（コンパイラ, インタプリタ）
 - 開発支援（プログラム解析, ツールなど）
 - 理論（意味論, 型システム, 検証など）
 - 参考：最近の研究で使っている言語（全部使えなければダメというわけではない）
 - OCaml, Rust, Haskell, C, C++, Nim, Erlang, Elixir, TypeScript, Scala, etc.
 - Coq, Promela (Spin)
- 低レイヤに興味のある人
 - OS, ミドルウェア, ランタイムシステム, デバイスドライバ