関数リアクティブプログラミング言語による小規模組込みシステムのプログラミング

東京工業大学 情報理工学院 情報工学系 渡部研究室

組込みシステムは，センサやスイッチ，通信ポートを介した外界からの入力に反応し，モータやディスプレイ等の機器を制御します．このような，外界からの入力に反応し続けるような動作をするシステムを**リアクティブシステム**と呼びます．組込みシステムの他にも，GUI，Webアプリケーション，ゲーム等はリアクティブシステムとみなすことができます．

一般にリアクティブシステムに与えられる入力は順序やタイミングが予測できません．そのため，ポーリングや割り込み，イベントシステム，状態機械と呼ばれる手法がリアクティブシステムのプログラミングにおいて現在一般的に用いられています．ところが，これらの手法を用いるとプログラムが分断されて非常に見通しが悪くなり，作成や保守のコストが増大します．

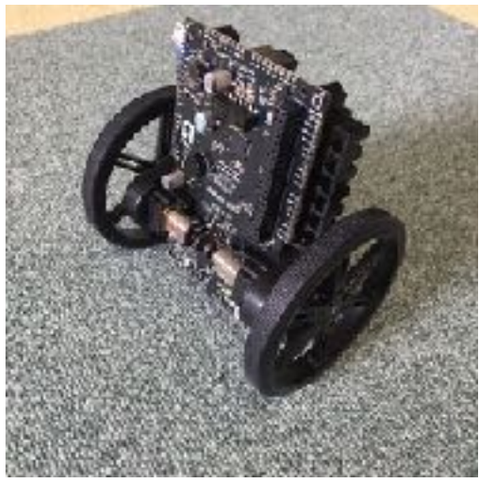
関数リアクティブプログラミング(FRP)は，副作用のない式のみを用いてリアクティブシステムのプログラムを記述しようという考え方です．これによってプログラムの見通しをよくし，保守コストを下げようという目的で研究が行われています．

関数リアクティブプログラミングによってリアクティブシステムの動作が見通しよく書けるのは，**時変値(Time Varying Value)**と呼ばれる仕組みによります．これは時々刻々と変化するデータをそのまま表現する機構です．例えば温度センサを表す時変値を temp とし，ファンのON/OFFを表す時変値を fan とします．気温が30度を超えたらファンを回すプログラムは以下の1行で書くことができます．

```
node fan : Bool = temp > 30
```

従来のプログラミング手法では，例えば気温の変化に追随するには，ポーリングによって温度センサによる測定を繰り返すか，あるいはセンサからの割り込みによって変化を知る必要がありました．FRPでは，センサの測定値やアクチュエータの状態を時変値として表現することでプログラムを簡潔に記述することができるようになります．

渡部研究室では，特にマイクロコントローラ等を用いた小規模組込みシステムを対象とした，FRPによる開発支援についての研究を行っています．これまでに，オリジナルのFRP言語Emfrp等の開発を通して，FRPが小規模組込みシステム開発に有用であることを明らかにしてきました．現在，並行計算モデルであるアクターモデルとFRPの融合による非同期性の表現と分散システムへの対応，文脈指向プログラミングの考え方によるモジュール性の強化，FRPによる実時間性の表現，および各種の応用について研究を行なっています．



倒立振子ロボット Pololu Balboa 32U4 ATmega 32U4 (32KB Flash, 2.5KB RAM)

FRPによる倒立振子ロボットの制御プログラムの例

```
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 StdL

...

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

node init[(0, 0)] (encL1, encR1) = (encL, encR)
node (speedL, speedR) = (encL1 - encL1@last, encR1 - encR1@last)
node init[(0, 0)] (distL, distR) = (distL@last + speedL, distR@last + speedR)

node risingAngleOff  = gyroY * angle_rate_ratio + angle
node fallingAngleOff = gyroY * angle_rate_ratio - angle

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

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

主な発表論文　T. Watanabe, "A Simple Context-Oriented Programming Extension to an FRP Language for Small-Scale Embedded Systems", COP 2018, ACM, Jul., 2018.
K. Sawada and T. Watanabe, "Emfrp: A Functional Reactive Programming Language for Small-Scale Embedded Systems", CROW 2016, pp. 46-54, ACM, Mar., 2017.