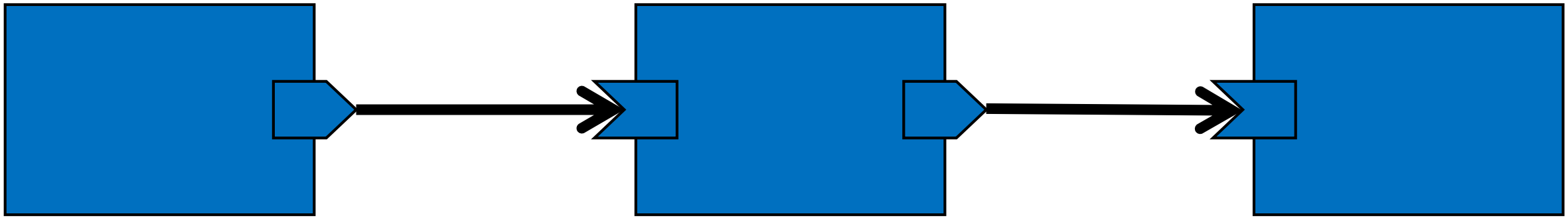
12-13コマ目
車輪移動ロボットの遠隔操作

SWジョイスティック動作プログラム作成

実習内容

- SWジョイスティックコンポーネントを車輪移動ロボット用コンポーネントに接続するコンポーネントを作成



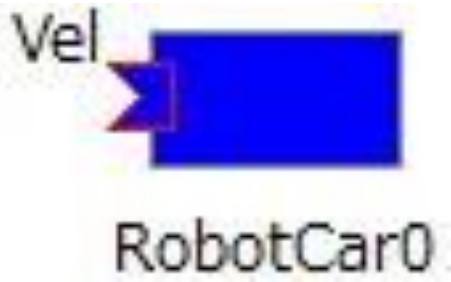
SWジョイスティックコンポーネント概要

コンポーネント名		
TkJoyStickComp		
概要		
GUIのスティックをドラッグで移動した値を出力		
ポート名	変数型	意味
pos	TimedFloatSeq	ジョイスティックのX-Y値
vel	TimedFloatSeq	対向2輪型移動ロボットの 各車輪の速度を出力 要素0に左側の速度， 要素1に右側の速度

スタートメニューから
[OpenRTM-aistx.y.z] →
[Python] →
[Components] →
[Examples]の
[TkJoyStickComp.py]で
起動する

車輪移動ロボット用コンポーネント概要

コンポーネント名		
RobotCar		
概要		
マウス型ロボットカーの制御コンポーネント		
ポート名	変数型	意味
Vel	TimedVelocity2D	車輪移動ロボットの速度を入力



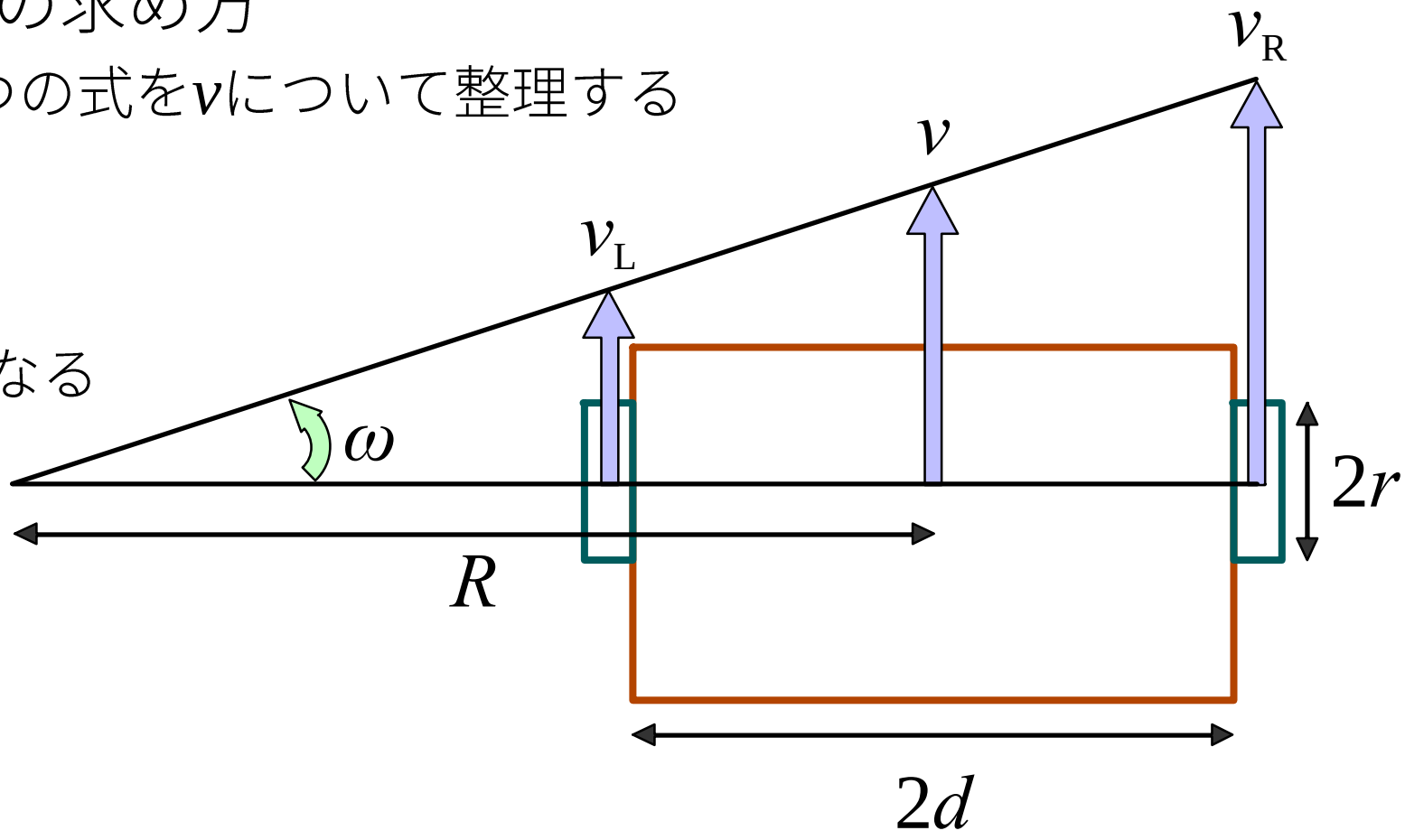
作成コンポーネント概要

コンポーネント名			
ConvertValue			
概要			
ジョイスティックの値を受け取り，マウス型ロボットカー用に値を変更			
ポート名	フローポート	変数	意味
VelIn	InPort	TimedFloatSeq	各車輪の速度を入力
VelOut	OutPort	TimedVelocity2D	2次元速度ベクトルを出力

ソフトウェアジョイスティックからの値は大きいため，
出力する際は，1/1000倍の値で使用する

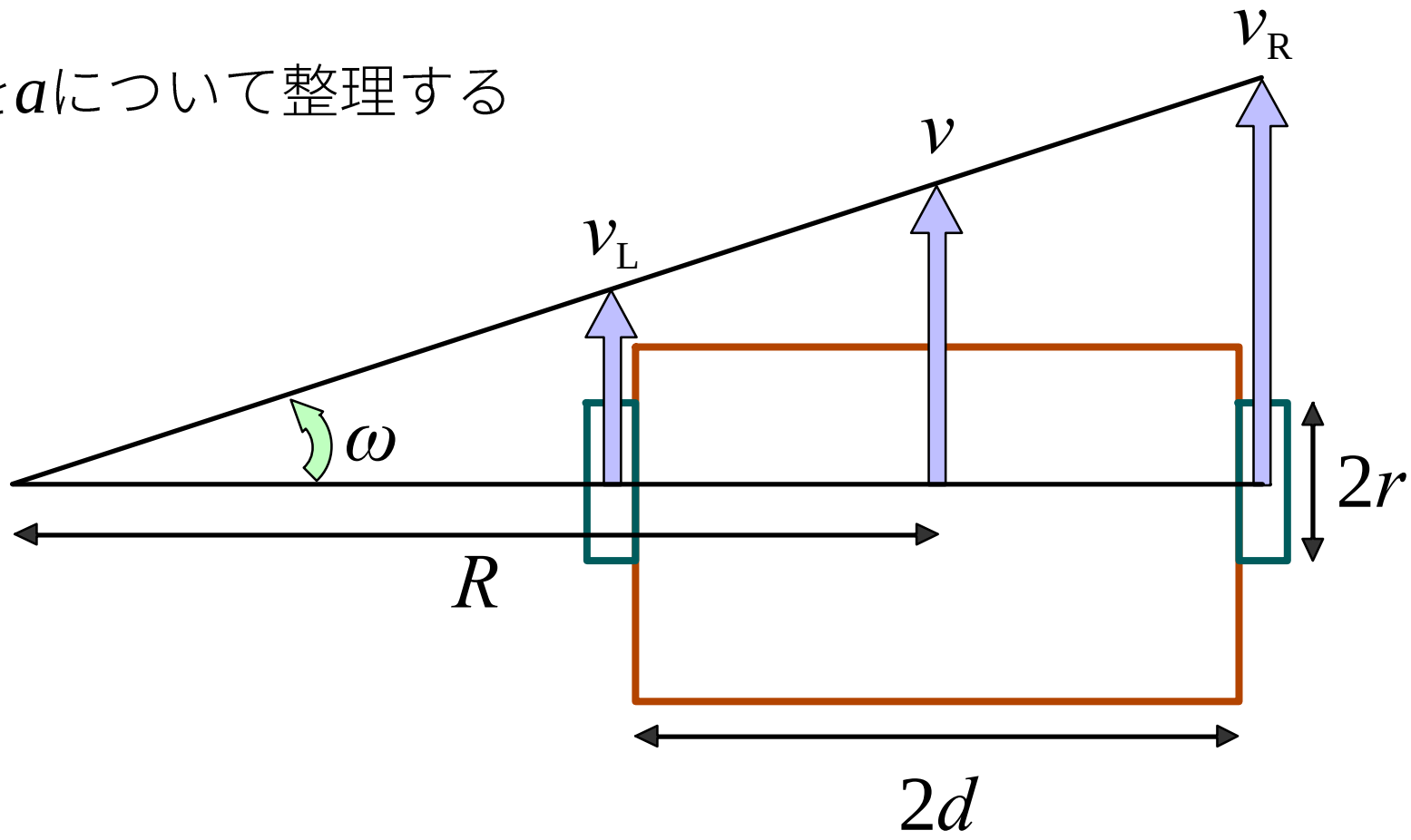
左右の車輪の速度から ロボットの速度と角速度への変換方法

- ロボットの中心の速度の求め方
  - 10コマ目で使用した2つの式をvについて整理する
    $v_L = v - d\omega$
    $v_R = v + d\omega$
  - まとめると以下の式となる
    $v = (v_L + v_R) / 2$



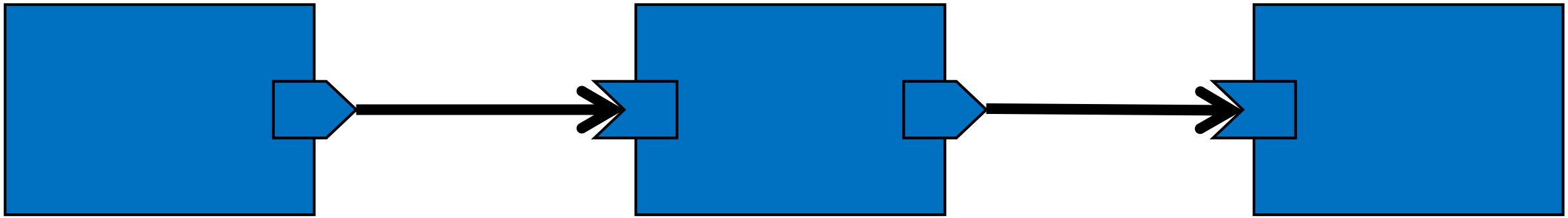
左右の車輪の速度から ロボットの速度と角速度への変換方法

- 角速度の求め方
  - 角速度は以下2つの式をaについて整理する
    $v_L = (R - d)\omega$
    $v_R = (R + d)\omega$
  - 以下の式となる
    $\omega = (v_R - v_L) / 2d$



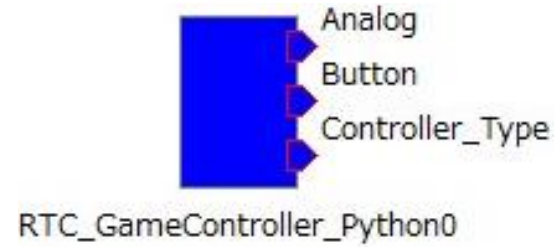
特別実習内容

- PS4コントローラ用コンポーネントから情報を受け取り，その情報に応じて制御値を計算して，車輪移動ロボット用コンポーネントに，その値を渡すコンポーネントを作成



PS4コントローラ用コンポーネント概要

コンポーネント名		
RTC_GameController_Python		
概要		
PS4コントローラのボタン状態を出力		
ポート名	変数型	意味
Analog	TimedDoubleSeq	アナログスティック値
Button	TimedULong	ボタンの押下状態
Controller_Type	TimedString	接続コントローラ名



https://rtc-fukushima.jp/component/2285/ からダウンロード

RTC.TimedDoubleSeq型

- RTC.TimedDoubleSeqは構造体で以下のように定義されている

データ型	変数名	意味
Double[]（リスト）	data	Double型のリスト
RTC.Time	tm	タイムスタンプ

- 使用例

```
self._d_Analog.data = [0.0, 0.0] # 初期化
self._d_Analog.data[0] = 2       # 値の代入
```

PS4コントローラ用コンポーネント

- アナログキー

PS4	値範囲	出力配列
十字キー↑↓	[-1 or 0 or 1] ↑押した場合：-1，デフォルト：0，↓押した場合：1	data[0]
十字キー←→	[-1 or 0 or 1] ←押した場合：-1，デフォルト：0，→押した場合：1	data[1]
左スティック上下	[-1, 1] 上に最も倒した時：-1，デフォルト：0，下に最も倒した時：1	data[2]
左スティック左右	[-1, 1] 左に最も倒した時：-1，デフォルト：0，右に最も倒した時：1	data[3]
右スティック上下	[-1, 1] 上に最も倒した時：-1，デフォルト：0，下に最も倒した時：1	data[4]
右スティック左右	[-1, 1] 左に最も倒した時：-1，デフォルト：0，右に最も倒した時：1	data[5]
L2	[-1, 1] デフォルト：-1，最大押し込み時：1	data[6]
R2	[-1, 1] デフォルト：-1，最大押し込み時：1	data[7]

PS4コントローラ用コンポーネント

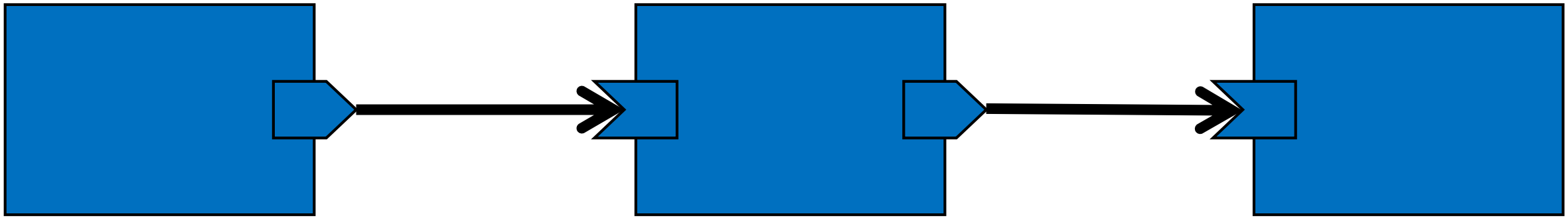
- ボタン
  - 押されたボタンの値を合計した数字が出力される
  - △ボタンを押した場合：8が出力
  - ◯ボタンとL３ボタンを同時に押した場合：4 + 1024 = 1028が出力

番号	1	2	3	4	5	6	7
PS4	□	×	○	△	L1	R1	L2
値	**1**	**2**	**4**	**8**	**16**	**32**	**64**
番号	8	9	10	11	12	13	14
PS4	R2	SHARE	Option	L3	R3	PS	**パッド**
値	**128**	**256**	**512**	**1024**	**2048**	**4096**	**8192**

実習解説

実習解説（11コマ目）

- SWジョイスティックコンポーネントを車輪移動ロボット用コンポーネントに接続するコンポーネントを作成



作成コンポーネントの内容

1. InPortからソフトウェアジョイスティック(RTC::TimedFloatSeq)の値を読み込む

2. 左右の車輪の速度から中心速度と角速度を計算

3. OutPortから中心速度と角速度(RTC::TimedVelocity2D)の値を出力

11コマ目 解答例

作成コンポーネント概要

コンポーネント名			
ConvertValue			
概要			
ジョイスティックの値を受け取り，マウス型ロボットカー用に値を変更			
ポート名	フローポート	変数	意味
VelIn	InPort	TimedFloatSeq	各車輪の速度を入力
VelOut	OutPort	TimedVelocity2D	2次元速度ベクトルを出力

コンポーネントの仕様

- RTCBuilderで以下のように設定

基本
- モジュール名：ConvertValue
- モジュール概要：任意(ConvertValue component)
- バージョン：1.0.0
- ベンダ名：任意
- モジュールカテゴリ：任意(Category)
- コンポーネント型：STATIC
- アクティビティ型：PERIODIC
- コンポーネントの種類：DataFlow
- 最大インスタンス数：1
- 実行型：PeriodicExecutionContext
- 実行周期：1000.0

選択アクションコールバック
- onInitialize
- onAcivated
- onExcute
- onDeactivated

InPort
- ポート名：VelIn
- データ型：RTC::TimedFloatSeq
- 変数名：inValue
- 表示位置：LEFT

OutPort
- ポート名：VelOut
- データ型：RTC::TimedVelocity2D
- 変数名：outValue
- 表示位置：RIGHT

言語・環境
- Python

プログラムの編集

プログラムの流れ

1. InPortからソフトウェアジョイスティック（RTC::TimedFloatSeq）の値を読み込む

2. RTC::TimedFloatSeqに格納されている左右のタイヤの速度から中心速度と角速度を計算

3. 中心速度と角速度をTimedVelocity2DのVxおよびVaに代入

4. 車輪移動ロボット用コンポーネントへ出力

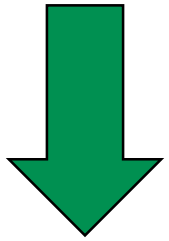
Python

- Pythonプログラムの編集手順
  - ConvertValue.pyの各関数を編集
    - __init__
      - 起動時によばれ，ポートの初期化を行う
    - onInitialize
      - コンポーネント起動時によばれるときに1度だけ初期化を行う
    - onActivated
      - 非アクティブ状態からアクティブ化されるとき1度だけよばれる
    - onDeactivated
      - アクティブ状態から非アクティブ化されるとき1度だけよばれる
    - onExecute
      - アクティブ状態時に周期的によばれる

__init__関数の変更

- __init__関数内のInPort，OutPortの初期化を以下のように変更する

```
self._d_inValue = RTC.TimedFloatSeq(*inValue_arg)
self._d_outValue = RTC.TimedVelocity2D(*OutValue_arg)
```



```
self._d_inValue = RTC.TimedFloatSeq(RTC.Time(0, 0), [])
self._d_outValue = RTC.TimedVelocity2D(RTC.Time(0, 0), RTC.Velocity2D(0.0, 0.0, 0.0))
```

onInitialize関数全文

- onInitialize関数に以下の1行を追加する

```
self.tread = 0.046 # ロボットの中心からタイヤまでの幅
```

- 以下のように追加する

```
def onInitialize(self):
    self.tread = 0.046 # ロボットの中心からタイヤまでの幅
    return RTC.RTC_OK
```

- 中心速度と角速度の計算に使用する

onActivated関数全文

- onActivated関数を以下のように編集する

```
def onActivated(self, ec_id):
      print "onActivated"
      return RTC.RTC_OK
```

onDeactivated関数全文

- onDeactivated関数を以下のように編集する

```
def onDeactivated(self, ec_id):
      print "onDeactivated"
      return RTC.RTC_OK
```

onExecute関数 | InPortからの値の読み込み

- InPortから値の読み込み

```
if self._VelInIn.isNew():
        # 値を読み込む
        data = self._VelInIn.read()
```

onExecute関数 | 中心速度と角速度の計算

- 中心速度と角速度の計算のプログラム

```
# 左車輪と右車輪の値から速度と角速度を求める
# 中央の速度を求める
Vx = (data.data[1] + data.data[0]) / 2
# 角速度を求める
omega = (data.data[1] - data.data[0]) / (2 * self.tread)
```

onExecute関数 | OutPortから値の出力

- コンポーネントから値を出力する

```
# 値が大きいため1000分の1にする
self._d_outValue.data.vx = Vx / 1000
self._d_outValue.data.vy = 0.0
self._d_outValue.data.va = omega / 1000
# 出力処理
self._VelOutOut.write()
```

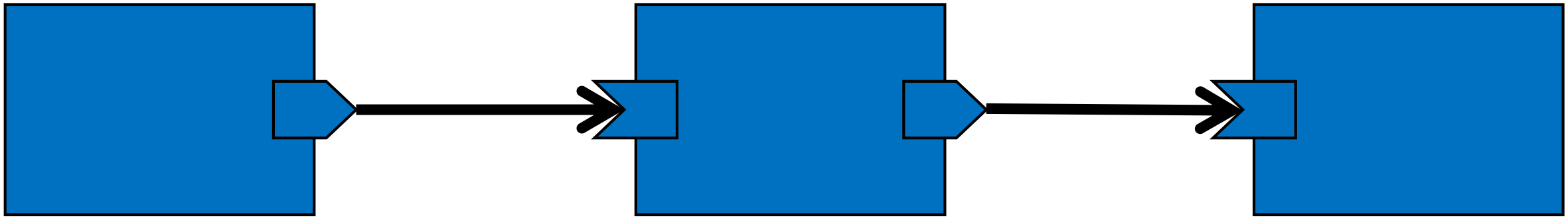
onExecute関数全文

- onExecute関数を以下のように編集する

```
def onExecute(self, ec_id):
        if self._VelInIn.isNew():
                # 値を読み込む
                data = self._VelInIn.read()
                # 左車輪と右車輪の値から速度と角速度を求める
                # 中央の速度を求める
                Vx = (data.data[1] + data.data[0]) / 2
                # 角速度を求める
                omega = (data.data[1] - data.data[0]) / (2 * self.tread)
                # 値が大きいため1000分の1にする
                self._d_outValue.data.vx = Vx / 1000
                self._d_outValue.data.vy = 0.0
                self._d_outValue.data.va = omega / 1000
                # 出力処理
                self._VelOutOut.write()
        return RTC.RTC_OK
```

実習解説（11コマ目 特別実習）

- PS4コントローラ用コンポーネントから情報を受け取り，その情報に応じて制御値を計算して，車輪移動ロボット用コンポーネントに，その値を渡すコンポーネントを作成



作成コンポーネントの内容

1. InPortからPS4コントローラのアナログ値(RTC::TimedDoubleSeq)を読み込む
   ※今回はアナログスティックの値を使用

2. アナログ値からロボット中心速度と角速度を計算

3. OutPortから中心速度と角速度(RTC::TimedVelocity2D)の値を出力

11コマ目 特別実習 解答例

作成コンポーネント概要

コンポーネント名			
GameController_Converter			
概要			
PS4コントローラのアナログスティック値の値を受け取り， マウス型ロボットカー用に値を変更			
ポート名	フローポート	変数	意味
Analog	InPort	TimedDoubleSeq	アナログスティック値を入力
vel	OutPort	TimedVelocity2D	2次元速度ベクトルを出力

コンポーネントの仕様

- RTCBuilderで以下のように設定

基本
- モジュール名：GameController_Converter
- モジュール概要：任意(GameController_Converter component)
- バージョン：1.0.0
- ベンダ名：任意
- モジュールカテゴリ：任意(Category)
- コンポーネント型：STATIC
- アクティビティ型：PERIODIC
- コンポーネントの種類：DataFlow
- 最大インスタンス数：1
- 実行型：PeriodicExecutionContext
- 実行周期：1000.0

選択アクションコールバック
- onInitialize
- onAcivated
- onExcute
- onDeactivated

InPort
- ポート名：Analog
- データ型：RTC::TimedDoubleSeq
- 変数名：Analog
- 表示位置：LEFT

OutPort
- ポート名：vel
- データ型：RTC::TimedVelocity2D
- 変数名：vel
- 表示位置：RIGHT

言語・環境
- Python

プログラムの編集

プログラムの流れ

1. PS4コントローラコンポーネントからの値を読み込む
2. アナログスティック値の値から中心速度と角速度を計算
3. 中心速度と角速度を車輪移動ロボット用コンポーネントへ出力

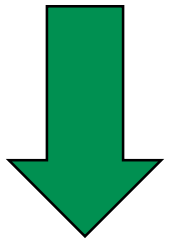
Python

- Pythonプログラムの編集手順
  - GameController_Converter.pyの各関数を編集
    - __init__
      - 起動時によばれ，ポートの初期化を行う
    - onActivated
      - 非アクティブ状態からアクティブ化されるとき1度だけよばれる
    - onDeactivated
      - アクティブ状態から非アクティブ化されるとき1度だけよばれる
    - onExecute
      - アクティブ状態時に周期的によばれる

__init__関数の変更

__init__関数内のInPort，OutPortの初期化を
以下のように変更する

```
self._d_Analog = RTC.TimedDoubleSeq(*Analog_arg)
self._d_vel = RTC.TimedVelocity2D(*OutValue_arg)
```



```
self._d_Analog = RTC.TimedDoubleSeq(RTC.Time(0, 0), 0)
self._d_vel = RTC.TimedVelocity2D(RTC.Time(0, 0), RTC.Velocity2D(0.0, 0.0, 0.0))
```

onActivated関数全文

- onActivated関数を以下のように編集する

```
def onActivated(self, ec_id):
      print "onActivated"
      return RTC.RTC_OK
```

onDeactivated関数全文

- onDeactivated関数を以下のように編集する

```
def onDeactivated(self, ec_id):
      print "onDeactivated"
      return RTC.RTC_OK
```

onExecute関数 | InPortからの値の読み込み

- InPortから値の読み込み

```
if self._AnalogIn.isNew():
        # 値の読み込み
        analog = self._AnalogIn.read()
```

onExecute関数 | 中心速度と角速度の計算

- アナログスティックの値を速度と角速度に変換するプログラム

```
# 車輪移動ロボットのTimedVelocity2Dの値に変換
# 計算式は試行錯誤で自分で考える
self._d_vel.data.va = analog.data[3] * -8.5 * 40 / 100
self._d_vel.data.vx = analog.data[2] * -0.5 * 40 / 100
self._d_vel.data.vy = 0
```

onExecute関数 | OutPortから値の出力

- コンポーネントから値を出力する

```
self._velOut.write() # 出力
```

- コンポーネントから値を出力するにはOutPortの変数に値を入力してwrite関数で出力する

onExecute関数全文

- onExecute関数を以下のように編集する

```
def onExecute(self, ec_id):
        if self._AnalogIn.isNew():
                # 値の読み込み
                analog = self._AnalogIn.read()

                # 車輪移動ロボットのTimedVelocity2Dの値に変換
                self._d_vel.data.va = analog.data[3] * -8.5 * 40 / 100
                self._d_vel.data.vx = analog.data[2] * -0.5 * 40 / 100
                self._d_vel.data.vy = 0

                # 出力
                self._velOut.write()

        return RTC.RTC_OK
```

プログラムコピー用ZIP

- 以下URLからコピー用のZIPファイルをダウンロード可能
  https://rtc-fukushima.jp/wp/wp-content/uploads/2018/11/12-13Program.zip
- 以下ファイルがあることを確認してください
  - ConvertToVel.txt
  - GameController_Converter.txt
- プログラムのコピーは上記ファイルから
  コピーするようにしてください