上級者向け講習会課題２

会津大学　ＲＴミドルウェア講習会

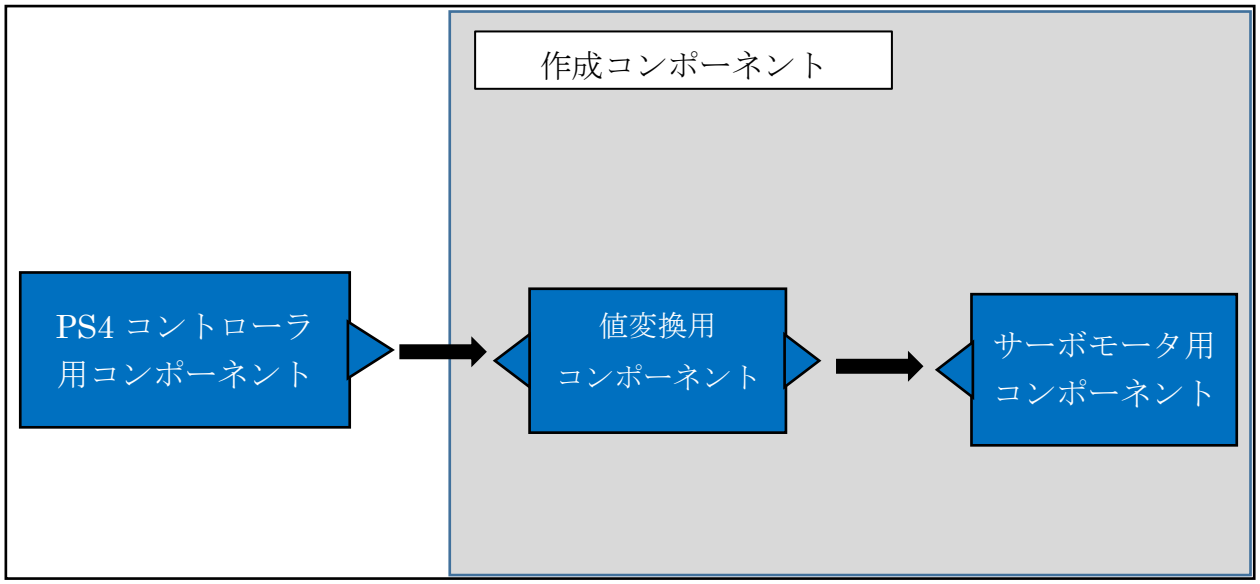
PS4 コントローラでロボットアームを動作させるシステムを作成

目次

1 課題概要

1.1 ロボットアームを操作するシステムの作成(課題 2)

PS4 コントローラでロボットアームを操作するシステムの作成。作成したら置いてある Raspberry Pi の箱を掴み、別の場所に置くという動作をしてみてください。



2 コンポーネント概要

2.1 サーボモータ用コンポーネント

コンポーネント名				
ServoMotorControl				
概要				
データポートから角度の値を受け取り、サーボモータを動かす				
ポート名	**フローポート**	**変数名**	**変数型**	**意味**
ServoMotor1	InPort	ServoMotorValue1	TimedFloat	FaBo PWM0 に接続しているサーボモータの値を入力
ServoMotor2	InPort	ServoMotorValue2	TimedFloat	FaBo PWM1 に接続しているサーボモータの値を入力
ServoMotor3	InPort	ServoMotorValue3	TimedFloat	FaBo PWM2 に接続しているサーボモータの値を入力
言語				Python

実際にプログラムをするとき、Python ではポート名と変数は以下の様に変化します。

InPort ポート名	InPort 変数名	OutPort ポート名	OutPort 変数名
self._ポート名 In	self._d_変数名	self._ポート名 Out	self._d_変数名

各データポートから値を受け取り、サーボモータを動作させるコンポーネントを作成します。
各データポートと対応するサーボモータは上記の通りです。

このコンポーネント単体の確認の時は以下のコンポーネントを使用してください。

上級者向け講習会課題 2

・ConsoleFloat

https://rtc-fukushima.jp/wp/wp-content/uploads/2017/11/SampleProgram.zip

中の ConsoleFloat.py を起動してウインドウから値を入力すると TimedFloat 型で値を出力します。これを作成したサーボモータ用コンポーネントに接続して動作確認をしてください。

2.1.1 注意事項

同じ値を連続でいれると故障の原因になるので注意してください。

従って値を入れる前には以下の様に if 文で前回の値と異なることを追加してください。

```
def calc_duty(self,angle):
        duty = 0.0
        duty = 102 + (491 - 102)/180 * angle
        return duty

def onExecute(self, ec_id):

        if self._ServoMotor1In.isNew():
                #前回の値保存#
                servo =self._d_ServoMotor1.data
                #今回のサーボモータ 1 値取得
                self._d_ServoMotor1 = self._ServoMotor1In.read()
                #今回の値が前回の値と異なることを確認
                if servo !=self._d_ServoMotor1.data:
                        #角度ど PWM 出力値に変換
                        duty_int= int(self.calc_duty(self._d_ServoMotor1.data))
                        self.bus.write_i2c_block_data(self.PCA9685_ADDRESS,6,[0, 0,
duty_int & 0xFF , duty_int >>8])
```

Python で新しい値が入っているか判定する関数と値を読み込む関数は以下になります。

新しい値が入っているか判定する関数	値を読み込む関数
isNew()	read()

2.2 値変換用コンポーネント

コンポーネント名				
ConvertToMotor				
概要				
PS4 コントローラからの値を受け取りサーボモータ用の値に変換して出力する				
ポート名	**フローポート**	**変数名**	**変数型**	**意味**
Analog	InPort	Analog	TimedDoubleSeq	PS4 コントローラからアナログスティックの値を受け取る
Button	InPort	Button	TimedULong	PS4 コントローラからボタンの押下状態を受け取る
ServoMotor1	OutPort	ServoMotorValue1	TimedFloat	サーボモータの値(角度)を出力
ServoMotor2	OutPort	ServoMotorValue2	TimedFloat	サーボモータの値(角度)を出力
ServoMotor3	OutPort	ServoMotorValue3	TimedFloat	サーボモータの値(角度)を出力

実際に C++でプログラムをするとき、ポート名と変数は以下の様に変化します。

InPort ポート名	InPort 変数名	OutPort ポート名	OutPort 変数名
m_InPort 変数名 In	m_変数名	m_OutPort 変数名 Out	m_変数名

PS4 コントローラ用コンポーネントから値を受け取り、サーボモータを動作させる値に変換して出力するコンポーネントを作成。

2.3 PS4 コントローラ

PS4 コントローラ用コンポーネントは以下の URL にあります。

https://rtc-fukushima.jp/component/1139/

コンポーネント名		
RTC_GameController_Win		
概要		
接続された PS4 コントローラのボタン押下状態とアナログ情報を出力		
ポート名	変数型	意味
Controller_Type	TimedString	接続デバイス名
Button	TimedULong	ボタン押下状態
Analog	TimedDoubleSeq	アナログ情報

・Button

ボタン押下時に押下したボタンの値を合計した数字が出力されます。

例 1)△を押された時：8 出力
例 2)○と L3 を同時に押したとき：4+1024=1028 で 1028 が出力される。

番号	PS4	値	番号	PS4	値
1	□	1	8	R2	128
2	×	2	9	SHARE	256
3	○	4	10	OPTION	512
4	△	8	11	L3	1024
5	L1	16	12	R3	2048
6	R1	32	13	PS	4098
7	L2	64	14	タッチパッドボタン	8196

上級者向け講習会課題 2

ボタンの値は合計なので確認するには、値を受けるコンポーネントで以下の方法などを使用して確かめることが出来ます。

```
long f = 1;
for (int i = 0; i < 14; i++)//1.各ボタンの値を格納
{
	binary_number[i] = f << i;
	std::cout << "bin num[" << i << "]:" << binary_number[i] << std::endl;
	button[i] = 0;
}
if (m_ButtonIn.isNew())//2.PS4 のボタンの情報が来ているか確認
{
	m_ButtonIn.read();//値読み込み
	button_input = m_Button.data;
	std::cout << "button_input:" << button_input << "b:";
	for (int i = 0; i < 14; i++)//3.どのボタンが押されているか確認
	{
		if (button_input >= binary_number[13 - i]){
			button[13 - i] = 1;//ボタンが押されている
			button_input = button_input - binary_number[13 - i];
		}
		else{
			button[13 - i] = 0; //ボタンが押されていない
		}
		std::cout << button[13 - i];
	}
	std::cout << std::endl;
```

1. 各ボタンの値を格納

binary_number の配列に各ボタンの値を格納しています。

binary_number[0]=1、binary_number[1]=2～binary_number[12]=4096、
binary_number[13]= 8196

2. PS4 のボタンが押されているか確認

m_ButtonIn.isNew()：新しい値が格納されているかを判定

m_ButtonIn.read()：新しい値を読み込み

3.　どのボタンが押されているか確認

ボタンの合計の値とボタンの値を比較し、合計の値以上の場合はボタンが押されていると判定。判定後合計の値からボタンの値を引く。合計の値より下の場合は押されていないと判断しています。

例えば×：2、R3：2048、タッチパッドボタン：8196 が押されている場合、合計は 10246 になります。すると以下の様な流れになります。

```
if 10246> 8196 → button[13]=1;10246- 8196
if 2050 < 4098 → button[12]=0;
if 2050 > 2048 → button[11]=1;2050-2048
if      2 < 1024 → button[10]=0;
if      2 < 0512 → button[9]=0;
if      2 <  256 → button[8]=0;
if      2 <  128 → button[7]=0;
if      2 <   64 → button[6]=0;
if      2 <   32 → button[5]=0;
if      2 <   16 → button[4]=0;
if      2 <    8 → button[3]=0;
if      2 <    4 → button[2]=0;
if      2 =    2 → button[1]=1;2-2
if      0 <    1 → button[0]=0;
```

この配列番号を上記表のボタン番号と照らし合わせると

14:タッチパッドボタン、12：R3、2：×が押されていると考えることが出来ます。

・Analog

Analog の値は以下の様になります。

PS4 コントローラ	値範囲	出力配列
十字キー↑ ↓	↑押した場合：-1、デフォルト：0、↓押した場合：1	data[0]
十字キー←→	←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	上傾き最大時：-1、デフォルト：0、下傾き最大時：1 -1～1 の間で値が変化	data[2]
左スティック右左	左傾き最大時：-1、デフォルト：0、右傾き最大時：1 -1～1 の間で値が変化	data[3]
右スティック上下	上傾き最大時：-1、デフォルト：0、下傾き最大時：1 -1～1 の間で値が変化	data[4]
右スティック右左	左傾き最大時：-1、デフォルト：0、右傾き最大時：1 -1～1 の間で値が変化	data[5]
L2	デフォルト：-1～押し込み最大：1 -1～1 の間で値が変化	data[6]
R2	デフォルト：-1～押し込み最大：1 -1～1 の間で値が変化	data[7]

3 PS4 コントローラ

3.1 PS4 コントローラについて

今回の遠隔操作では PS4 コントローラを使用します。
PS4 コントローラは MicroUSB か Bluetooth で PC に接続することが出来ます。
今回は MicroUSB で接続します。

3.2 PC との接続方法

PS4 コントローラの後ろの部分に USB(Micro-B)を差し込み、USB(A)を PC の USB ポートに差し込みます。ドライバーがインストールされれば使用出来る様になります。

3.3 設定方法

PS4 コントローラの設定を変更する場合は以下の手順で行ってください。(デフォルトのでよい場合は変更する必要はありません。)

1) [スタートメニュー]->[コントロールパネル]->[ハードウェアとサウンド] ->[デバイスとプリンター]の順で進んでください。

2) [Wireless Controller]のアイコンがあるのでそれを右クリックし[ゲームコントローラの設定]を選択してください。

3) 新しいウインドウが出ましたら[プロパティ]を選択。プロパティのウインドウが出たら、[調整]のタブを選択。そして[調整]を選択して変更してください。