・Ros node 構造

・topic の命名

・動作全体像

・制御 node 内部の status 定義

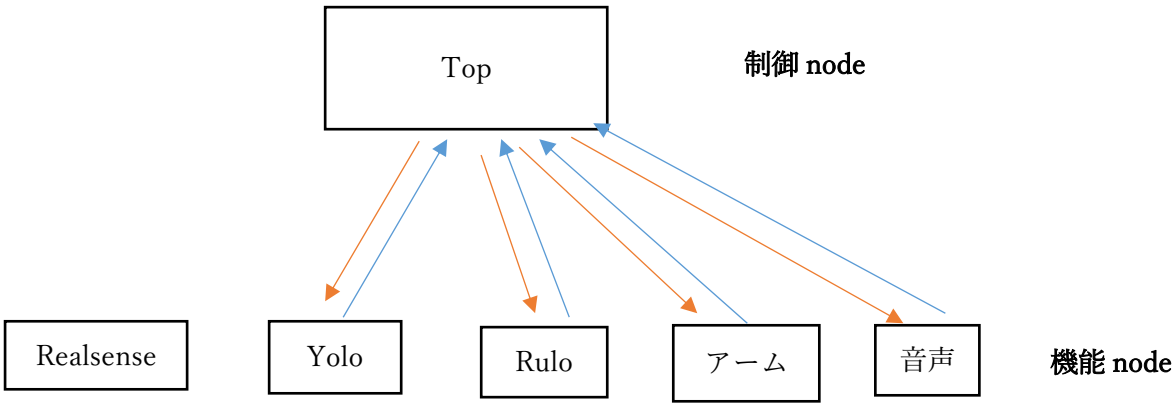
・topic message の構造と定義

・サンプルコード

・デモシーンの実現（一部）

・役割分担

・Ros　node 構造



結果 topic

指示 topic

＃topic の msg が共通で定義

・topic の名前

結果 topic：各 node により異なります

Yolo: results_YOLO, Rulo: results_RULO，アーム: results_ARM，音声: results_VOICE

指示 topic

Command_TOP

＃各 node を上記の topic 名前の受信と発行を設定してください

・動作全体像

各機能 node が Top からの指示 topic を受信して、実行した結果を結果 topic で送信

デモの動作制御（status の遷移など）は全部制御 node の中で処理する

各機能 node について、デモの動作状態を一切考慮せず制御 node からの指示 topic を待ち

機能 node は下記の 2 種類状態しかない

実行：実行→結果 topic 送信→次の指示 topic 待ち

待ち：次の指示 topic 待ち

＃詳細が以降の page のサンプルコードを参照

・status 定義

制御 node しか使わない。機能 node に関係ない

Status0: 初期　　必要な定義や callback 関数の登録とか

Status1: 待機

Status2: 探索

Status3: 移動

Status4: 微調整

Status5: 把持

Status6: PR

上記の status が TOP の作成により変更する可能性があります

機能 node に影響をしないので、各機能 node の実装には上記の status に依存しない

・結果 topic/指示 topic の msg 構造

../msg/common.msg

Common.msg	
Int	topic_id
NODE	node
RESULTS	ret
MSG	msg

Topic の Msg 構造

NODE	
ALL=0	//全 node
Yolo=1	
Rulo=2	
arm=3	
voice=4	
invalid=5	

指示 topic の場合は、どの node 向けの指示
結果 topic の場合は、無意味（TOP がこの値を参照しない）

RESULTS
Ok=0
Ng=1
ERR_code_1=2
ERR_code_2=3
…
Invalid=..

指示 topic の場合は、無意味
結果 topic の場合は、各 node の実行結果の状況を設定
＃ERR_code_xx は debug 用　各自で必要に合わせて追加すれば

MSG	
Float32 x3d, y3d, z3d	//三次元位置座標
Int32 x2d, y2d	//二次元位置座標
Int32 id	//class id / rulo 動作仕方
Int32 key	//keyboard command

Yolo 関係
Id=0 タバコ
Id=1 人

rulo 関係
Id=0 ランダム移動
Id=1 目的地に移動
Id=2 pose の微調整

指示 topic の場合は、機能 node の動作に必要な情報を設定
＃例えば、rulo に三次元位置を渡す、rulo が三次元位置に移動する
＃例えば、yolo に id を渡す、yolo が id の class だけを認識します
結果 topic の場合は、実行結果の詳細を設定
＃例えば、yolo の場合は認識した三次元や二次元の結果を設定

・サンプルコード

#image を伝えるためのサンプルなので、ビルドが通らないと思う
#同じ機能が実現できれば、別に下記の書き方じゃなくても OK

機能 node(YOLO)

```
void mainCallBack (const common::ConstPtr& msg)
{
  if(msg->node == 1 || msg->node == 0)   //TOP から実行指示受ける
    {
      YOLO のコア処理       //YOLO の認識処理　既にコードある
      common results;        //結果 topic msg を定義
      results.topic_id = msg->topic_id;
      results.ret = OK;      //実行結果の状況を設定　認識できた→OK
                             //認識できなかった→NG
                             //何かエラーがあれば、エラーコードを設定
      results.msg.x3d=…      //認識結果を設定
      results.msg.y3d=…
      results.msg.z3d=…
      publish(results);      //結果 topic を発行
      return;                //次の指示 topic を待つ
    }
  Else                       //他の node 向けの指示なので、yolo が次の指示 topic を待つ
    {
      return;                //次の指示 topic を待つ
    }
}
```

機能 node(Rulo)

```
void mainCallBack (const common::ConstPtr& msg)
{
  if(msg->node == 2 || msg->node == 0)   //TOP から実行指示受ける
    {
     Switch(msg->msg.id)   //動き方が msg.id の値を参照
       case 0:
          ランダム移動              //既にコードある？
          common results;          //結果 topic msg を定義
          results.topic_id = msg->topic_id;
          results.ret = OK;         //実行結果の状況を設定
          break;
       case 1:
          目的地への移動          //msg.x3d などにより目的地を設定既にコードある？
          common results;          //結果 topic msg を定義
          results.topic_id = msg->topic_id;
          results.ret = OK;         //実行結果の状況を設定
          break;
       case 2:
          pose 微調整               //msg.x2d y2d により pose を微調整　既にコードある？
          common results;          //結果 topic msg を定義
          results.topic_id = msg->topic_id;
          results.ret = OK;         //実行結果の状況を設定
          break;
       publish(results);
       return;                    //次の指示 topic を待つ
    }
  Else                          //他の node 向けの指示なので、次の指示 topic を待つ
    {
      return;                   //次の指示 topic を待つ
    }
}
```

機能 node(arm)

```
void mainCallBack (const common::ConstPtr& msg)
{
  if(msg->node == 3 || msg->node == 0)   //TOP から実行指示受ける
    {
      アームのコア処理      //把持処理   #既にコードある？
      common results;        //結果 topic msg を定義
      results.topic_id = msg->topic_id;
      results.ret = OK;       //実行結果の状況を設定   動作完了→OK
      publish(results);       //結果 topic を発行
      return;                 //次の指示 topic を待つ
    }
  Else                      //他の node 向けの指示なので、次の指示 topic を待つ
    {
      return;                //次の指示 topic を待つ
    }
}
```

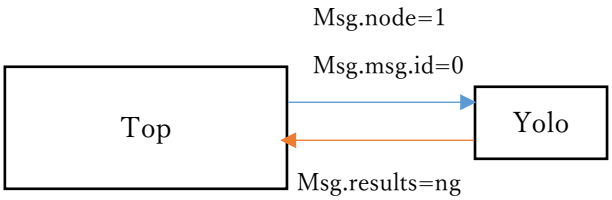
制御 node(TOP)

```
void mainCallBack (const common::ConstPtr& msg)
{
  State machine (省略)
  common msg
  msg.node = 1;   //yolo を動かせ
  msg.msg.id = 1; //人を検知して
  pusblish(msg);
}
```

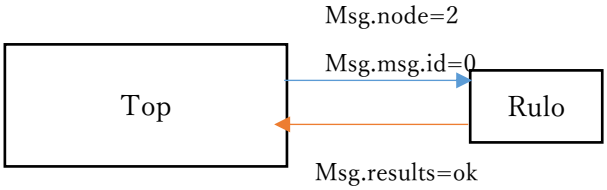
・デモシーンの一部

TIME

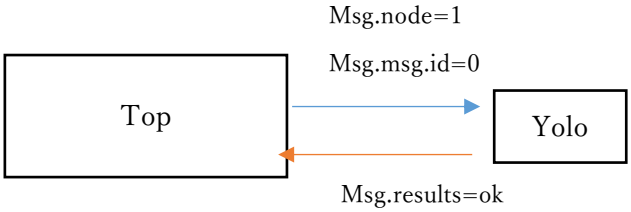
動作：タバコを見つけて、近づいて隠す



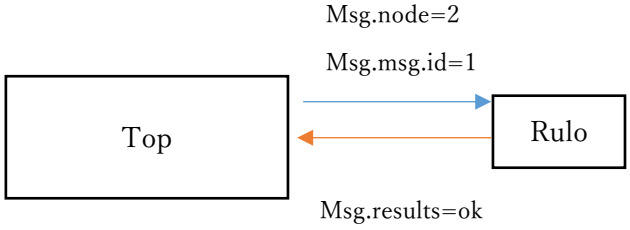
「タバコを認識する」指示を yolo に出す
Yolo が認識できなかった結果 topic を発行



「ランダム移動」指示を rulo に出す
Rulo が移動完了した結果 topic を発行

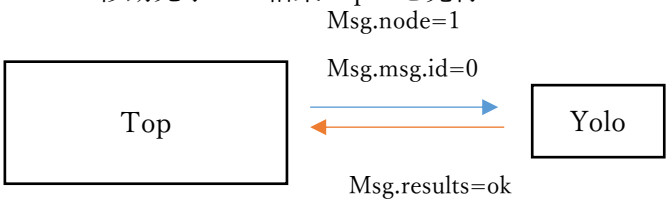


「タバコを認識する」指示を yolo に出す
Yolo が認識できた結果を topic 発行



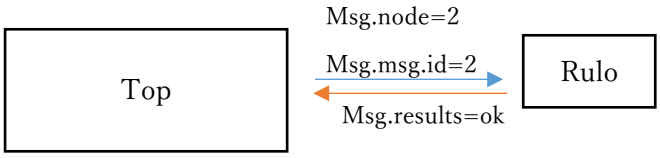
「目的地に移動」指示を rulo に出す

Rulo が移動完了した結果 topic を発行



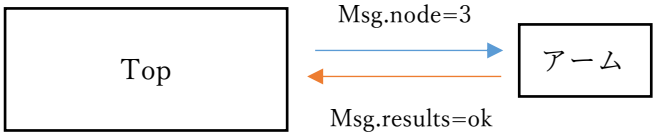
「タバコを認識する」指示を yolo に出す

Yolo が認識できた結果 topic を発行



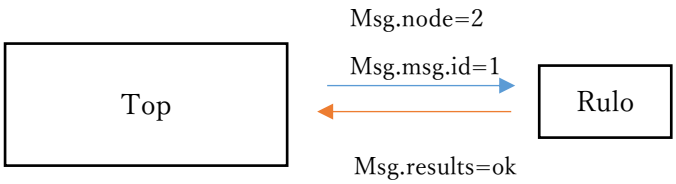
「pose 微調整」指示を rulo に出す

Rulo が動作完了した結果 topic を発行



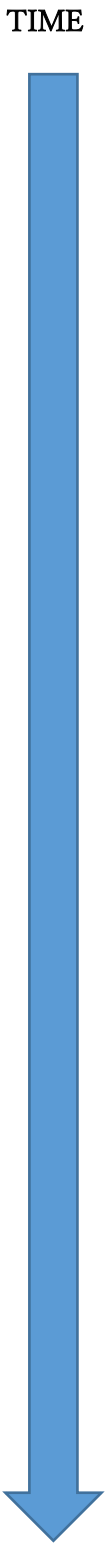
「把持」指示をアームに出す

アームが同さん完了した結果 topic を発行



「目的地に移動」指示を rulo に出す

Rulo が移動完了した結果 topic を発行



・役割分担

TOP（制御 node）　→　居
Yolo　→　居
アーム　→　長田さん　濱田さん
音声　→　長田さん　濱田さん
Rulo　→　長田さん
という感じかな？