Developing for the OpenRTM Based Software Controller

○ Yuichiro Kawasumi,
Hajime Saito(General Robotix,Inc.),
Noriaki Ando(AIST),
Rosen Diankov(University of Tokyo)

2Q1
IS:Advances in Open-source RoboticsTools(1/2)

Talk Overview

Development of motion control system on OpenRTM.

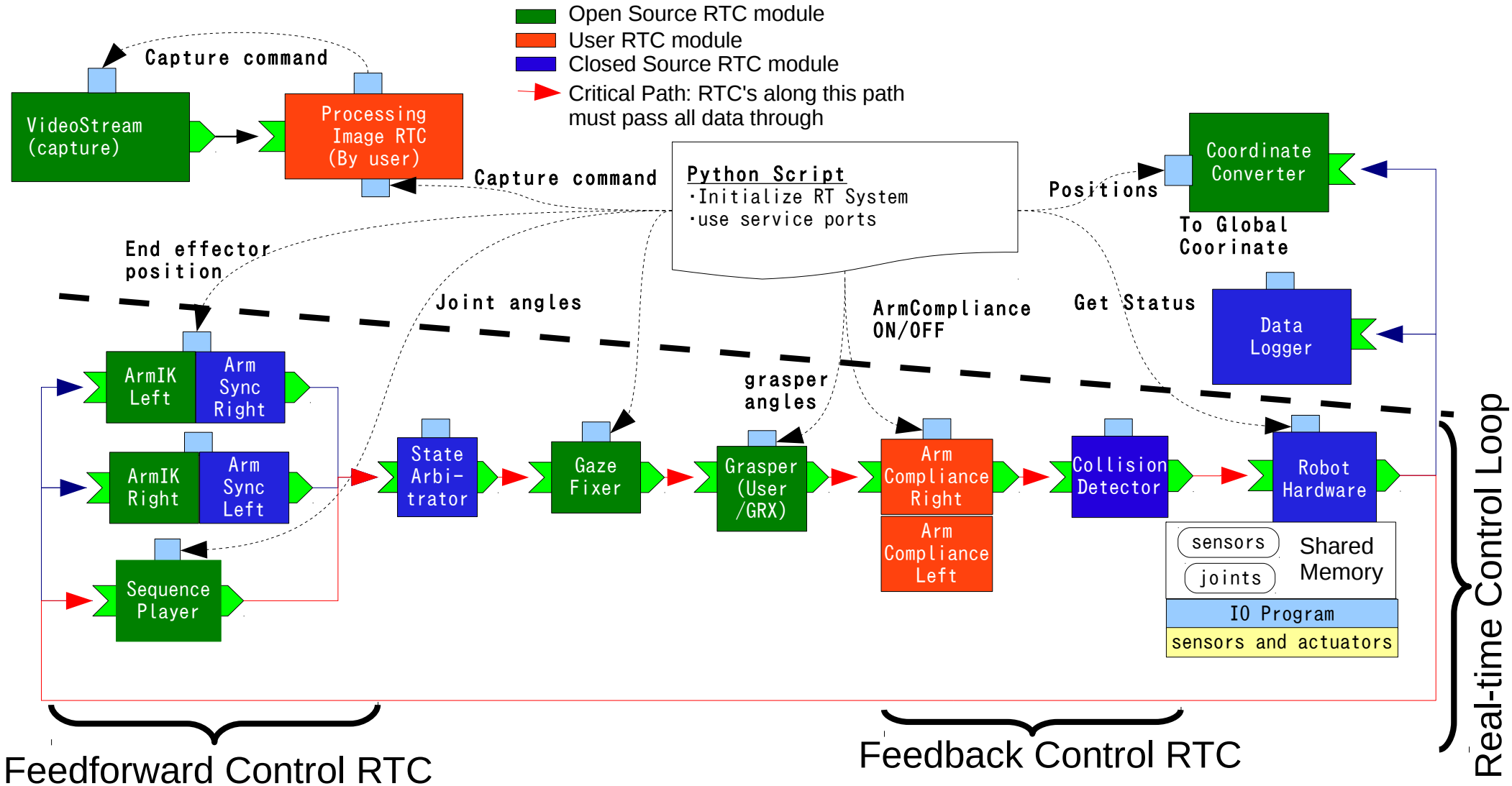
- Target Platform
- System Overview
- Motion Control System, Basic Philosophy
- Implementation of RTC

Target Platform
- Dual arm robot “HIRO”

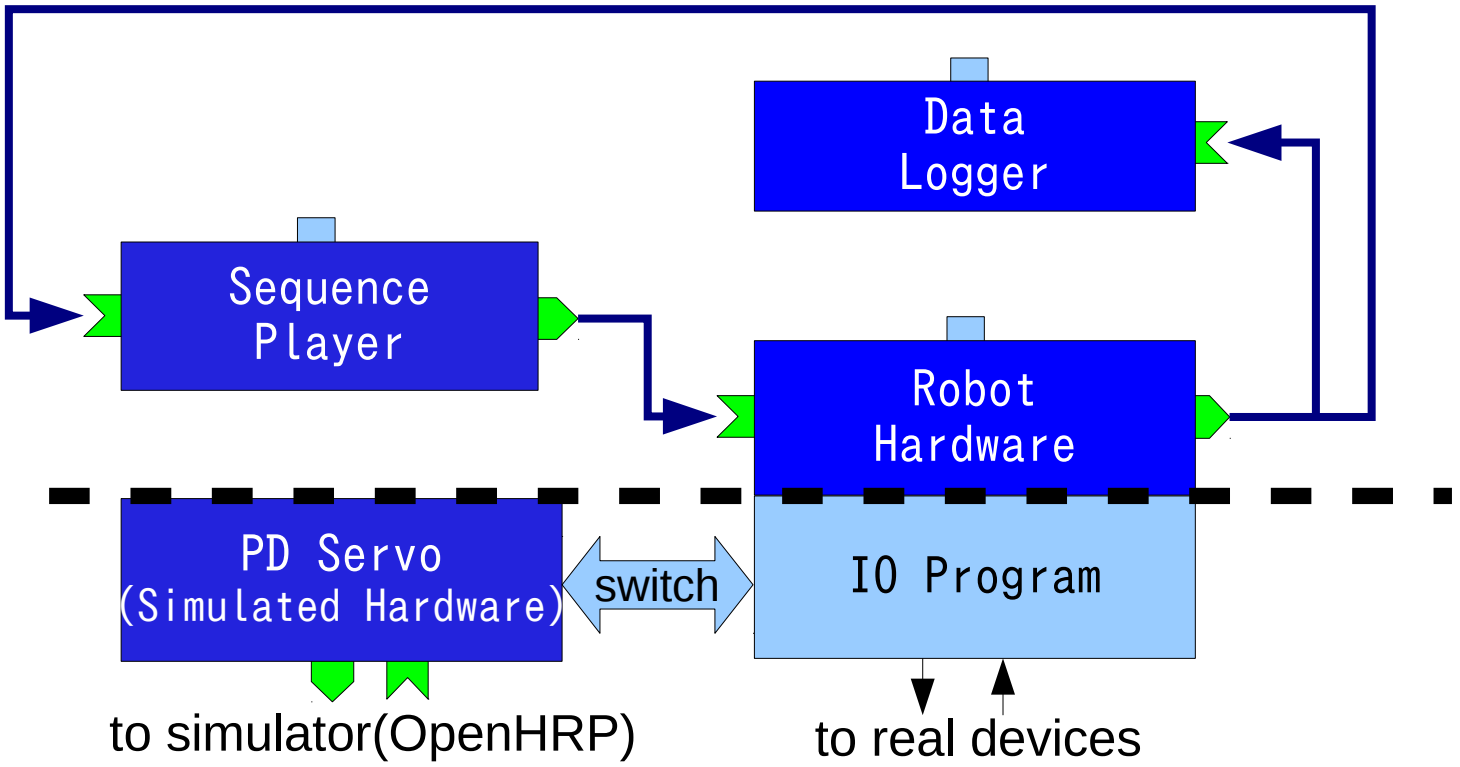
URL: http://nextage.kawada.jp/index.html

- 23 DOF
  - Head 2 DOF
  - Waist 1 DOF
  - Arm 6 DOF
  - Hand 4 DOF
- 4 usb cameras
  - 2 on head
  - 2 on each hand

System Overview



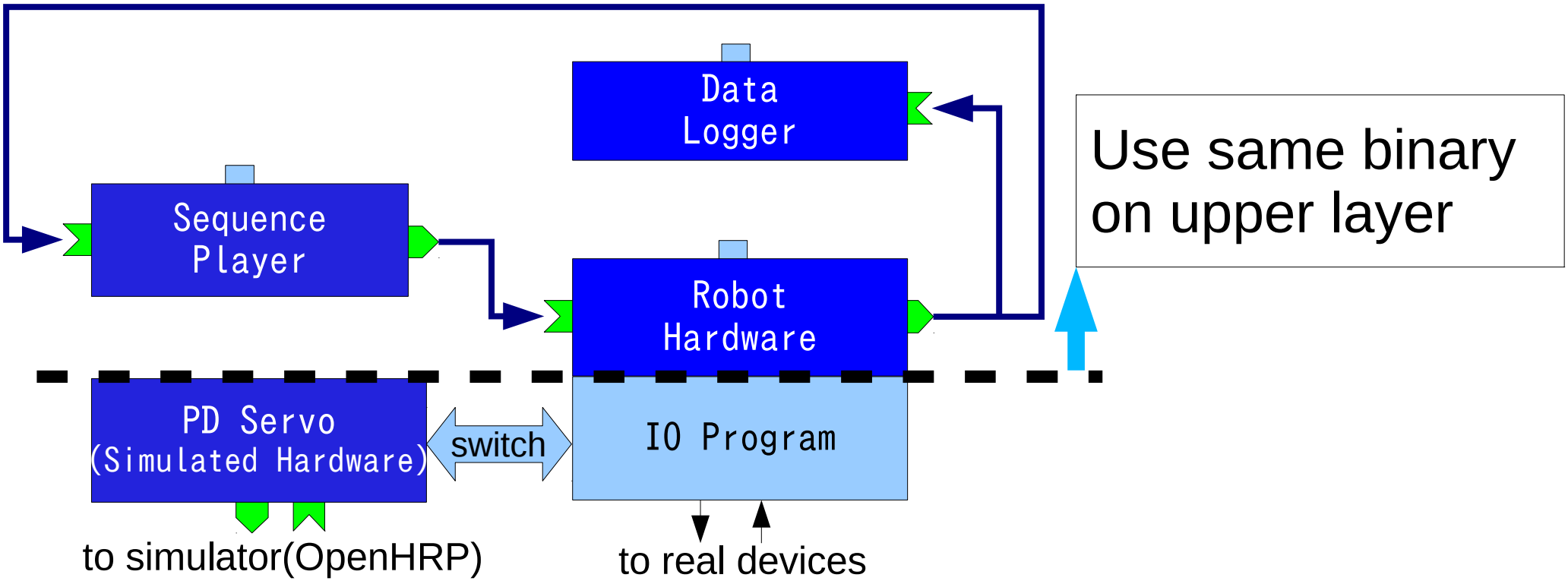
System Overview (Simplified)



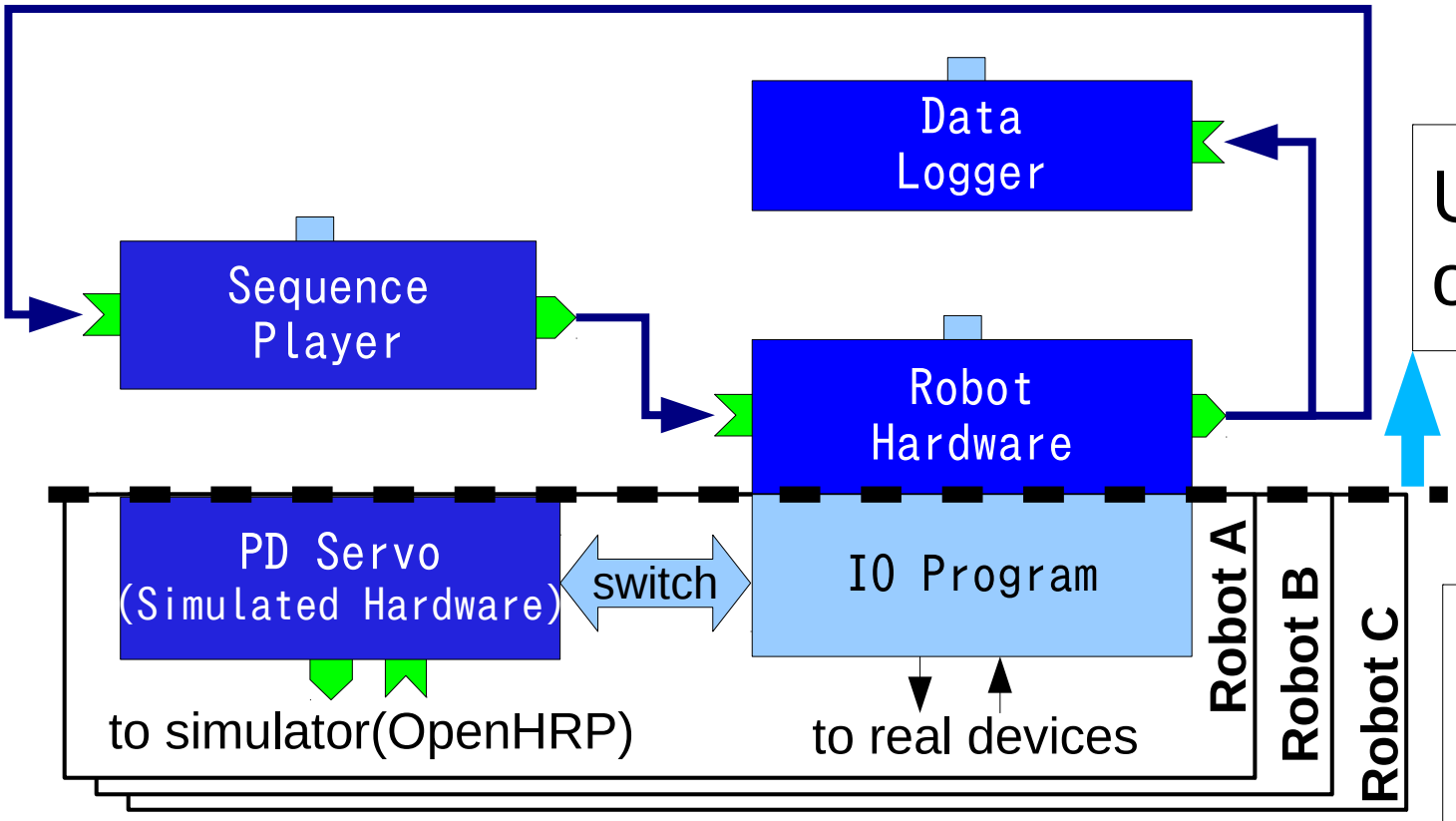
Motion Control System, Basic Philosophy

- Binary compatibility between Simulation, Actual Robot
- Common component structure(identical I/O port)
  - Base class for Component Implementation
- Python(or Jython) operation interface

Binary Compatibility simulation and real robot



Binary Compatibility simulation and real robot(s)

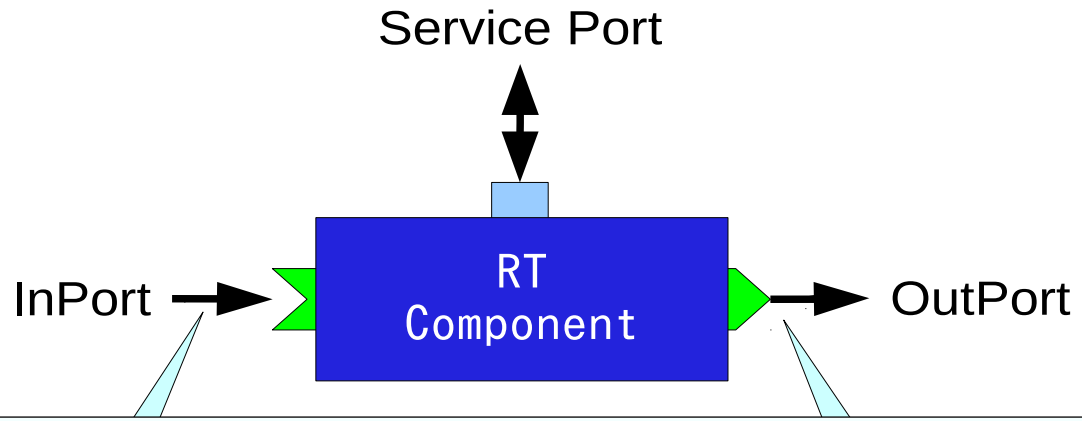


Use same binary on upper layer

We are preparing lower layer for various robots.

Common component structure

Identical I/O port




```
struct TimedJointData
{
    Time tm;                // the time at which data created
    sequence<Time> cmd;     // the time at which command
    sequence<octet> id;     // ID of RTC
    sequence< sequence<double> > qCommand;
    sequence< sequence<double> > dqState;   // actual velocity
    seque
    seque
    sequence<octet> collisionFlag;          // collision status
};
```

For motion arbitration

Joint State

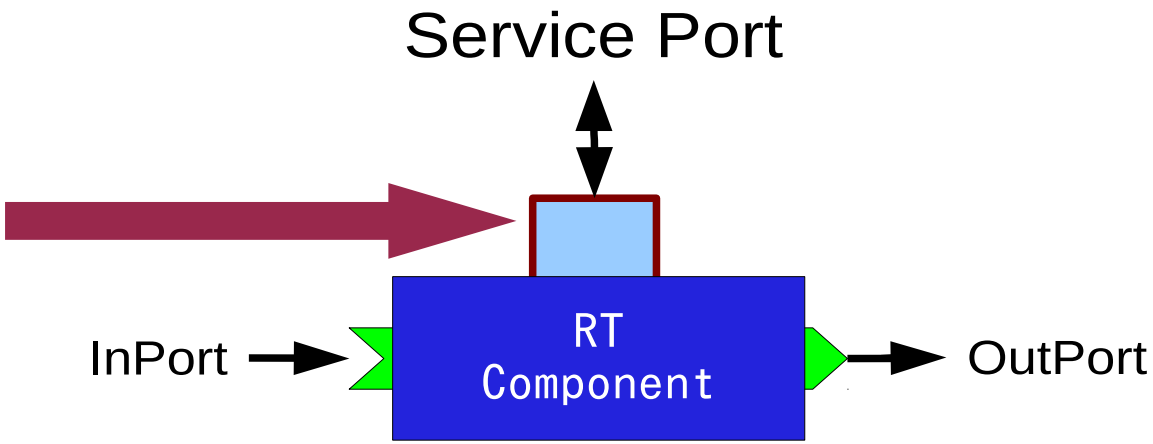
→Joint values for each part(BODY, RARM, RHAND, ...)

Service Port for Interface

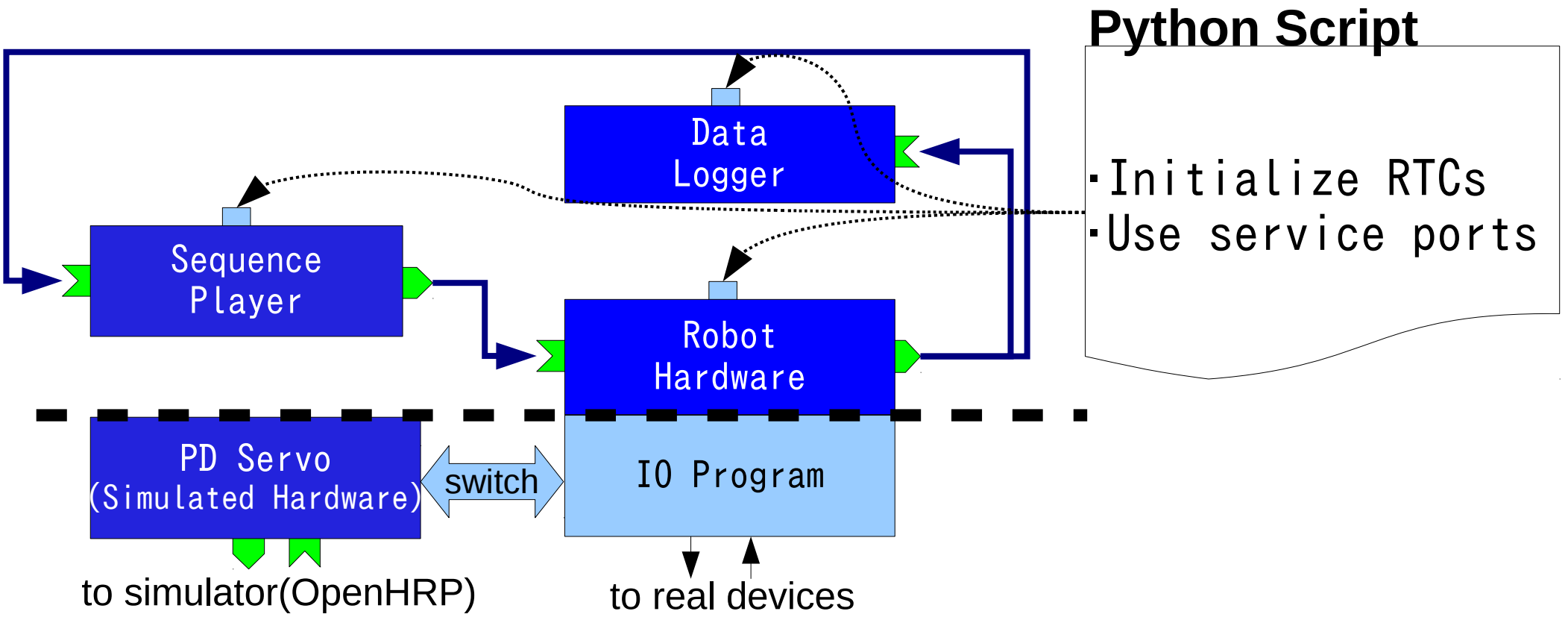
To communicate from external client,

1. define the interface as idl(Interface Definition Language)
2. generate c++ template code by idl compiler
3. implement the service functions
4. compile all of them and incorporate into the RTC

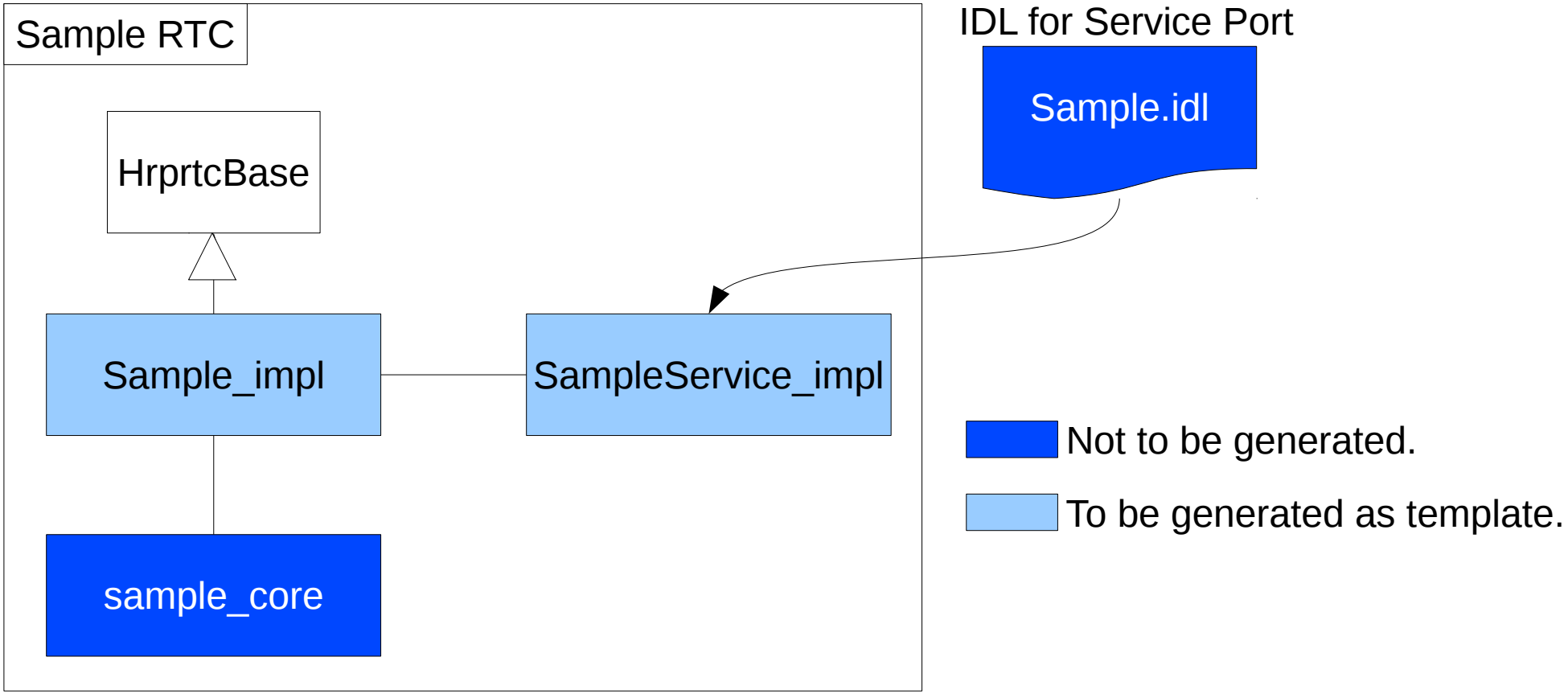
```
#Sample.idl

interface SampleService
{
        void hello();
};
```



Python operation interface



Implementation of RTC
Overview of Class Structure



Implementation of RTC Base Class (HrprtcBase)

By Common functions you can use ...

- Calculate with robot model
- the limit values for each joint
- map of joint id to part id (RARM, LARM, etc).

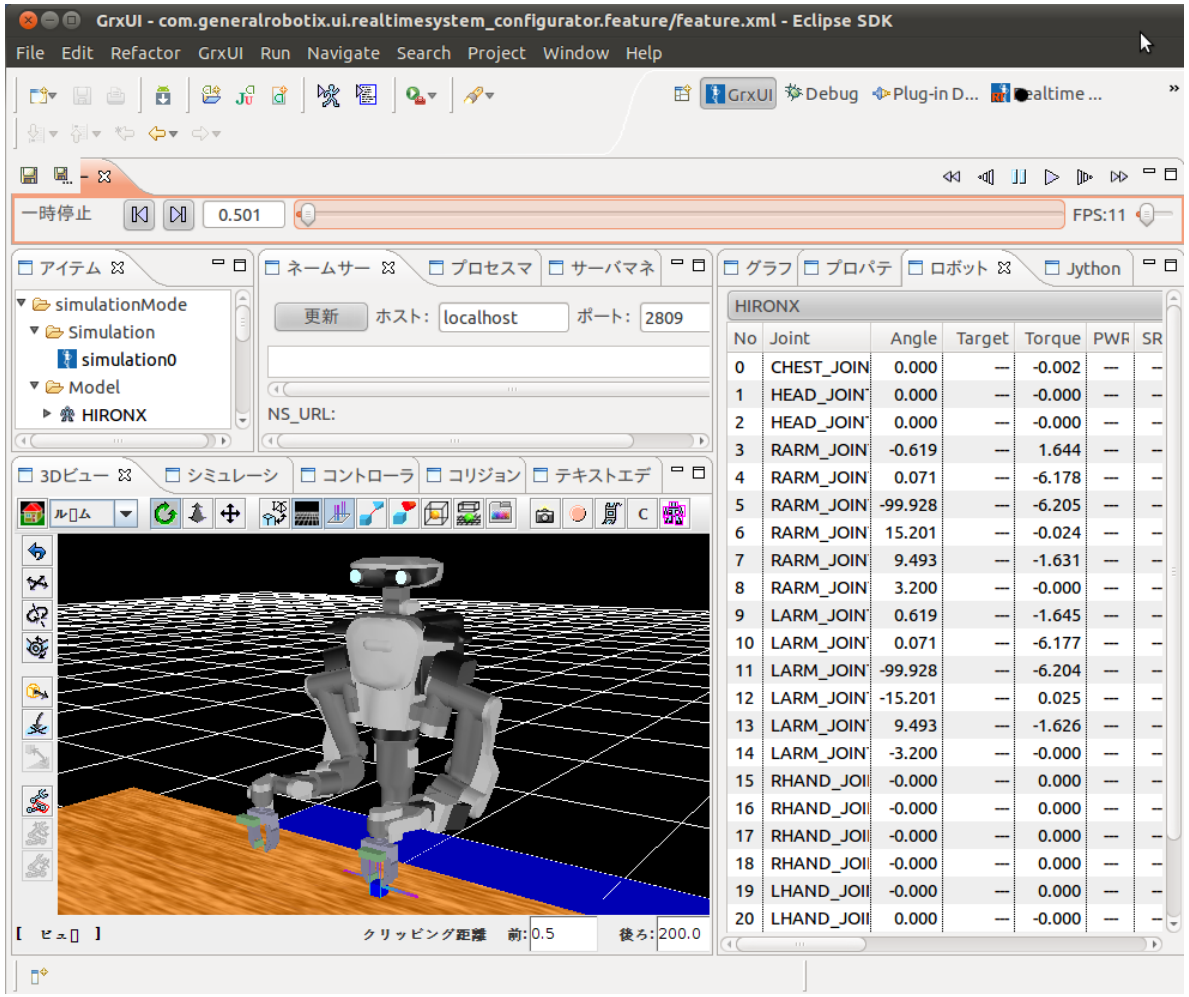
Implementation of RTC
Fill the template functions

```
ReturnCode_t  Sample::onInitialize() {
  HrprtcBase::onInitialize(); // <---- model and part info. is initialized
  /* implement other Initialization */
}

ReturnCode_t Sample::onExecute(RTC::UniqueId ec_id) {
  m_jointDatIn.read(); // <--- read joint data from InPort
  /* implement  main process */
  m_jointDatOut.write(); // <--- write joint data to OutPort
}
```

To Compile using CMake

```
#CmakeLists.txt
...
set(comp_name Sample)
set(comp_sources ${comp_name}.cpp)
set(comp_sources ${comp_sources} HrprtcBase.cpp)
set(comp_sources ${comp_sources} sample_core.cpp)  ---> add your codes
...
generate_stub_skel(${comp_name}Service)
generate_hrpsys_stub_skel_wrapper(JointDataTypes)
...
target_link_libraries(${comp_name} yourlibrary)  ---> add your libraries
...
```

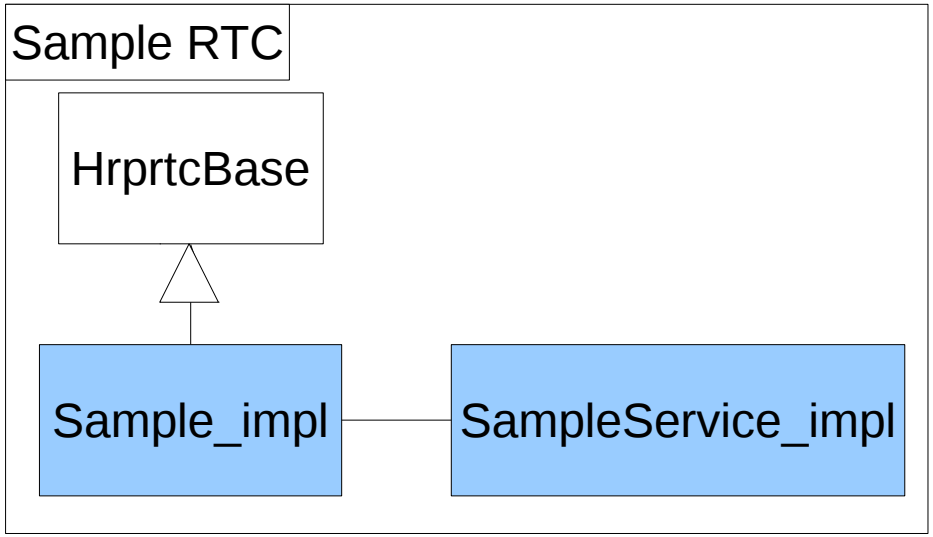
Test with OpenHRP



Summary

- We introduce the robot control system for dual arm robot 'HIRO'.
- We separate common functions as super class for motion control RTC, to create RTC easily for user.
- You can get them from google code soon.
- Latest infomation is available from googlegroups:

  “propelopenrtm”



Environment & Resouces

- OS:
  - Ubuntu Linux 10.04 LTS (32bit, 64bit)
  - QNX 6.5
- Open Source Projects on google code:
  - openhrp-aist-grx (openhrp with collada model)
  - hrprtc-grx-video-stream (capturing by usb camera)
  - hrprtc-grx-arm-ik(Ik sample using OpenRAVE) => not yet!
  - hrprtc-grx-sample (template sample) => not yet!
- Forum
  - Google groups : propelopenrtm