OROCOS- Easy way to Develop Open Source Robot Control Application

Tarlika Machhi[1], Naveen Rastogi[2], Pramit Dutta[2],Prerna Paliwal[3]

[1]Student, M.Tech (VLSI & Embedded Systems) U.V.Patel Collage of Enginerring, Mehsana, Gujarat, India
[2]Engineer SC, Institute for Plasma Research, Bhat, Gandhinagar, Gujarat, India
[3]Technical Associate, Device Driver, eiTRA, Ahmedabad, Gujarat, India

---***---

Abstract -*Industrial robotics and automation systems should be developed in such a way that they can be used for multiple purpose. Controlling of these robots and machines required to be done in real time. These type of control systems are combination of both hardware and software. Controlling software plays an important role in controlling hardware. It is required to develop an application which can fulfil user requirements like real time controlling, flexibility, reusability, maintainability, lower cost etc. The concept of middleware plays very strong role in the entire software development. It is required to choose an appropriate software platform or framework for developing application which can fulfill major of requirements of robotics. Using Open Robot Control Software (OROCOS) toolchain, we can develop architecture independent hard real time application for robot and machine control by implementing kinematic model. The OROCOS platform allows applications to be built as highly configurable and interactive component-based real-time control applications. The Real-Time toolkit (RTT) allows components to run on (real-time) operating systems and offers real-time scripting capabilities. So OROCOS play major role in developing controlling application for robots and machines.*

KeyWords: **Component, Deployer,FSM, KDL, OROCOSOCL, RTT**

1. INTRODUCTION

This paper is aimed to be a guiding paper on Orocos overview and Orocos application creation. Open Robot Control Software provides open source, platform independent, architecture independent hard real time component based framework for application development for robot and machine control. It provides four libraries which includes major of the functionalities for making a complete real time robot controlling application. An OROCOS component is a basic unit that executes one or more tasks, which are determined by its activity. Those tasks are functions in the C or C++ language or, a script in its own language called Orocos Programming Script (.ops) or a hierarchical state machine file (.osd). [5]Section II describes the basics of OROCOS toolchain libraries and the overview of Orocos Component .It includes basic component functionalities and states. Section III describes the Application development by creating component for TCP Client and deploying them into our application. Section IV includes the results for application and Section V concludes the paper.

2. OROCOS TOOLCHAIN OVERVIEW

2.1 OROCOS LIBRARIES

The **Real-Time Toolkit (RTT)** library: allows application designers to build highly configurable and interactive component-based real-time control applications. Component framework is provided as TaskContext Class.

The **Orocos Components Library (OCL)**: provides some components models for general purpose. Deployer component is used to load our components in application. TaskBrowser provides console window to give commands to the component.

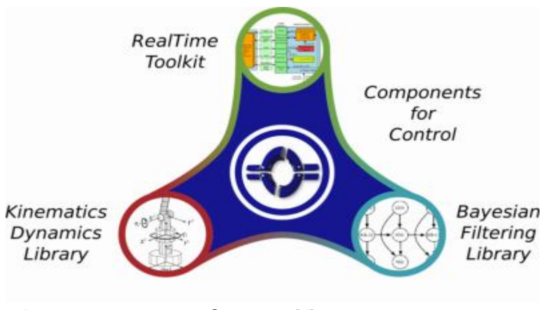


Fig -1: OROCOS Libraries[5]

The **Orocos Kinematics and Dynamics Library (KDL)**: allows for the calculation of kinematic chains in real-time. Robotic hardware is described and implemented as KDL Chain and it provides kinematic and dynamic solvers, and inbuilt functions related to kinematics and dynamics.

The **Orocos Bayesian Filtering Library (BFL)**: an independent framework for inference in Dynamic Bayesian Network.[5]

2.2 OROCOS COMPONENT HOOKS

Orocos applications are developed by composing software components. Software components can be from OCL library and also can be implemented by own using Orocos Real-Time Toolkit. Components are implemented by using Task Context class. It defines component interface, properties, attributes, operations, services, its peer components .It uses its Execution Engine for executing its programs and processing asynchronous messages. Component can be started only after getting configure call executed successfully. A function called as user 'hook' is available for each API function as shown in Fig-2.

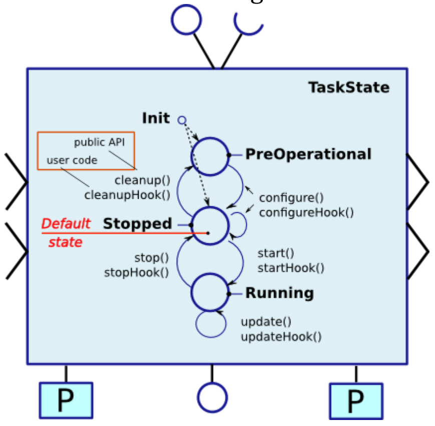


Fig -2: OROCOS Component States & Hooks[3]

configureHook () is for reading xml file and printing status messages. Functions for component configuration are called in this hook function.
startHook () starts the component.
updateHook (): Functions for doing work while component is in running state are called in this hook. Component is continuously updating after specific defined period. Period can be set using setPeriod(), by default it is zero.
stopHook (): Component can be stop manually or automatically by calling stop()
cleanupHook (): Write xml ,free resource. After completing the process cleanup is done by this hook and component returns into preoperational state.[3]

2.3 Component States

Orocos component has specific states**.**
Init: A component is in the Init state during its creation.
PreOperatinal: After instance of component is created, component enters the PreOperational or Stopped (default) state.
Stopped: after configuration component goes into stopped state. It is default state of component when it is loaded.
Running: After starting component by start (), it goes in running state and continuously runs updateHook ().
Here, tasks performed by component is started by object of class Activity. Activity class takes parameters like Period which shows execution time for periodic tasks. UpdateHook gets executed periodically by this period. Priority parameter is used to set Priority when multiple activities needs to be executed in single application. Scheduler parameters decides scheduling algorithm in which activities gets scheduled. It takes two value either ORO_SCHED_RT which is used for a real time scheduler or ORO_SCHED_OTHER, which is used for non-real time scheduler.[3][4][5]

3. APPLICATION DEVELOPMENT

3.1 Component Creation

Component package can be created by using command **orocreate-pkg component/package name**, which will create a complete component package. After that using cmake, it is built and then using make install we can compile the component package and install its library file (.so) in specific directory by assigning directory path to cmake flag DCMAKE_INSTALL_PREFIX. By exporting this path as RTT_COMPONENT_PATH, we can make these component libraries available in our environment for importing them in our application. When the component is build and install, shared object (.so) library is created, which can be used by Deployer Component.[3]

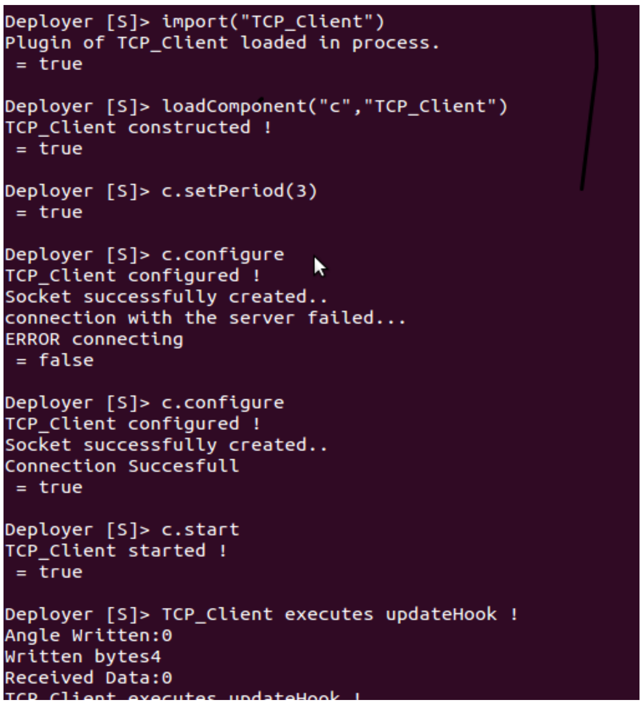
3.2 Component Deployment

Deployer component from OCL Library is used for running Orocos Application, which uses TaskBrowser Component which provides browser or terminal like windows for giving data and command to the Deployer and components. Here using the Deployer the Component are imported form the folder and instance of that component is created. Using the basic ops commands component is configured, started and stopped.

4. RESULTS AND DISCUSSION

Here for basic component creation, component named **TCP client** is created using Orocos toolchain for controlling purpose over distributed network. This component acts as client, which will provides angle data to the remote server. Remote server receives data and control the hardware accordingly. Transmission Control Protocol (TCP) is used for data transfer over network. TCP server is created on raspbian Operating system, installed on Raspberry Pi 2 Model B, which is connected to the servomotor of the robotic arm.
Import command will import the component package library folders in Deployer. By using **loadComponent** specific instance of the component named c of component type TCP Client is created as a peer of deployment component. Then in PreOperational mode configuration of the component is done. Then when we start the component, update hook is

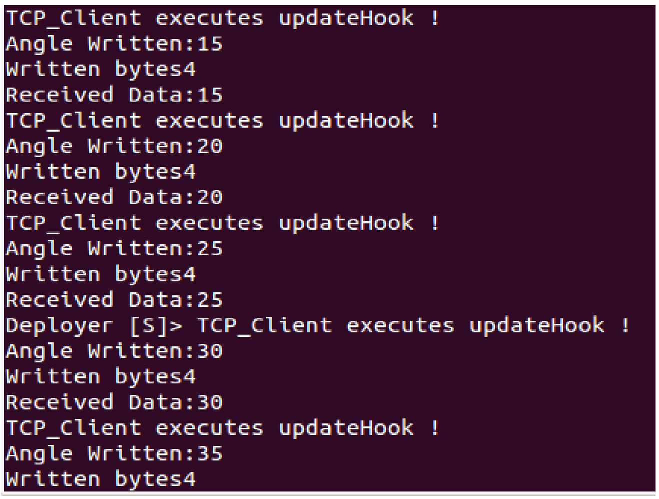
executed periodically. We can set the period manually in our application.



```
Deployer [S]> import("TCP_Client")
Plugin of TCP_Client loaded in process.
 = true

Deployer [S]> loadComponent("c","TCP_Client")
TCP_Client constructed !
 = true

Deployer [S]> c.setPeriod(3)
 = true

Deployer [S]> c.configure
TCP_Client configured !
Socket successfully created..
connection with the server failed...
ERROR connecting
 = false

Deployer [S]> c.configure
TCP_Client configured !
Socket successfully created..
Connection Succesfull
 = true

Deployer [S]> c.start
TCP_Client started !
 = true

Deployer [S]> TCP_Client executes updateHook !
Angle Written:0
Written bytes4
Received Data:0
TCP_Client executes updateHook !
```

Fig- 3 Orocos Component Deployment

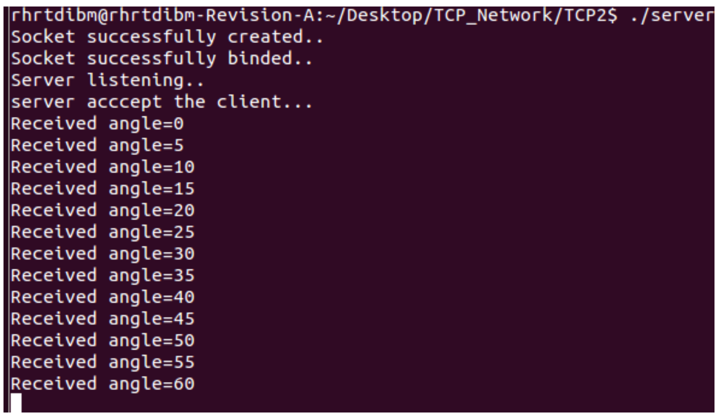
Also we can stop the component. Here component state is set to PreOperational so that it is in unconfigured state so before starting it we need to configure it first. In configuration, connection related functionalities are added here.



```
TCP_Client executes updateHook !
Angle Written:15
Written bytes4
Received Data:15
TCP_Client executes updateHook !
Angle Written:20
Written bytes4
Received Data:20
TCP_Client executes updateHook !
Angle Written:25
Written bytes4
Received Data:25
Deployer [S]> TCP_Client executes updateHook !
Angle Written:30
Written bytes4
Received Data:30
TCP_Client executes updateHook !
Angle Written:35
Written bytes4
```

Fig-4 Period Execution of Orocos Component

If sever is running and connection gets success then configuration gets succeeds otherwise fails.
It sends angle by incrementing it by 5 periodically. Period is set to 3 second by using setPeriod(3).



```
rhrtdibm@rhrtdibm-Revision-A:~/Desktop/TCP_Network/TCP2$ ./server
Socket successfully created..
Socket successfully binded..
Server listening..
server acccept the client...
Received angle=0
Received angle=5
Received angle=10
Received angle=15
Received angle=20
Received angle=25
Received angle=30
Received angle=35
Received angle=40
Received angle=45
Received angle=50
Received angle=55
Received angle=60
```

Fig- 5 Server Execution

Server takes angle from TCP Client and use is for controlling servomotor attached to its GPIO pin. Servo moves 5 degree at every 3 second as period is set at TCP Client. After stopping the Deployer, Component cleanup is done automatically by Deployer and then space allocated by Deployer is freed.

5. CONCLUSION

Using OROCOS Toolchain libraries, it is easier to develop real time controlling application for robotic hardware. Reusability of components provides great benefits for application developer as once library for specific component is created user can import it and then configure it according to its requirements without changing code and compiling them again. For applications which needs to perform tasks periodically, RTT has provides great functionality of getting component task execution periodically and changing period at run time. By developing control application using OROCOS, it saves time period for changing code and recompiling it again and again by providing reusability of components

REFERENCES

[1] Darlan Ioris, Walter Fetter Lages, Diego Caberlon Santini, “Integrating the OROCOS framework and the barrett wam robot”, 5th Workshop in applied robotics and automation, RobotControl 2012, UFRGS, Porto Alegre, Brazil, (2012), Source: www.ece.ufrgs.br

[2] Hurwitz, J. *Open Robot Control Software: the OROCOS project*. Proceedings of the 2001 IEEE International Conference on Robotics & Automation, Seoul, Korea.

[3] Component Building, “Orocos-component-manual”,from people.mech.kuleuven.be

[4] Orocos Component Library, “Orocos-ocl-intro”, from people.mech.kuleuven.be

[5] Orocoas Orocos Installation, “Orocos-user-manual”, frompeople.mech.kuleuven.be

[6] OROCOS,”OpenRobot Control Software”, www.orocos.org