ROS and CodeSys communication

Mahmoud Omar Ouali

Introduction

In this document I will illustrate four architectures of communication between ROS and CodeSys. For our project we are required to make a communication bridge between the CodeSys system and the Robot Operating System.

System Overview

The PC-based motion controller (MH2) from Delta Electronics is used to be the controller, MH2 is a SoftPLC CodeSys based. The PC-based motion controller is a powerful embedded computer with Intel Celeron J1900 processor, 4GB of RAM. MH2 is running windows 10 IoT and a Real time extension CodeSys Control RTE of 3S-Smart software Gmbh to ensure the realtime requirement of the CodeSys system[1].

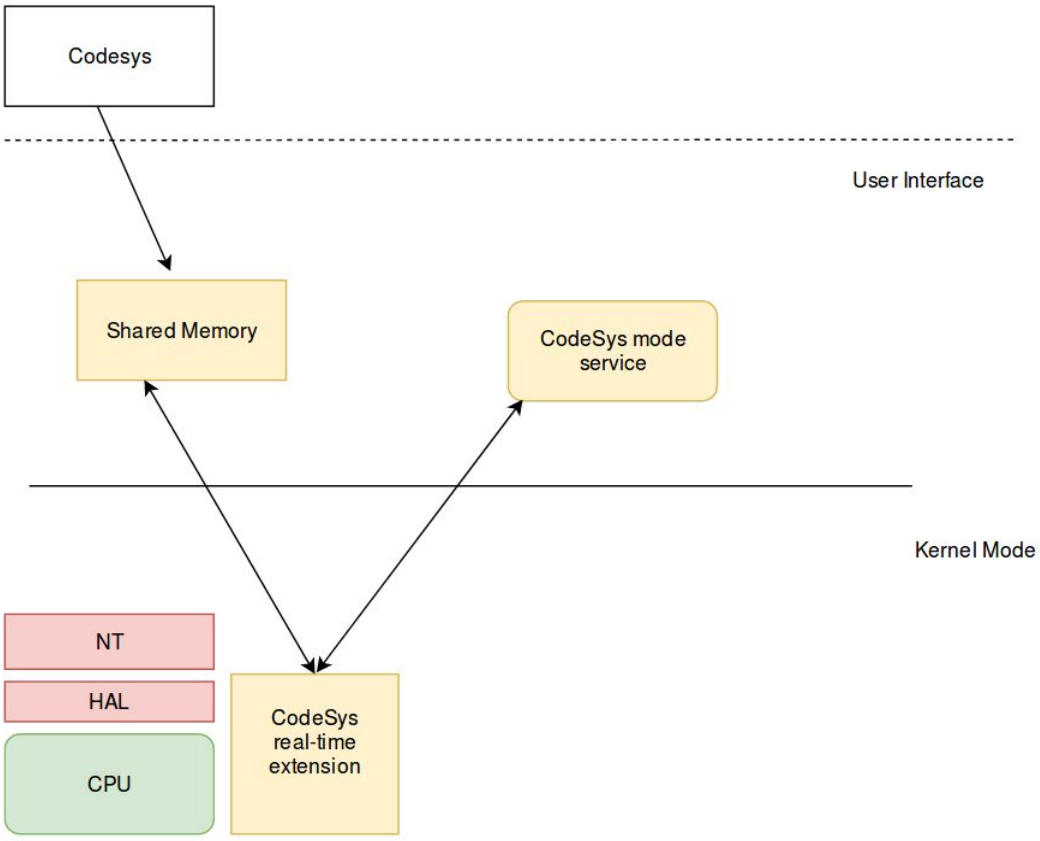

Figure1: The controller overview

As illustrated in figure 1, to ensure the real time requirement for CodeSys system, CodeSys control realtime extension (RTE) has being installed on the machine.
The RTE is running on two physical cores of the CPU, while the other two cores are reserved for the windows IoT.

Communication Architectures

1.ROS and CodeSys on MH2

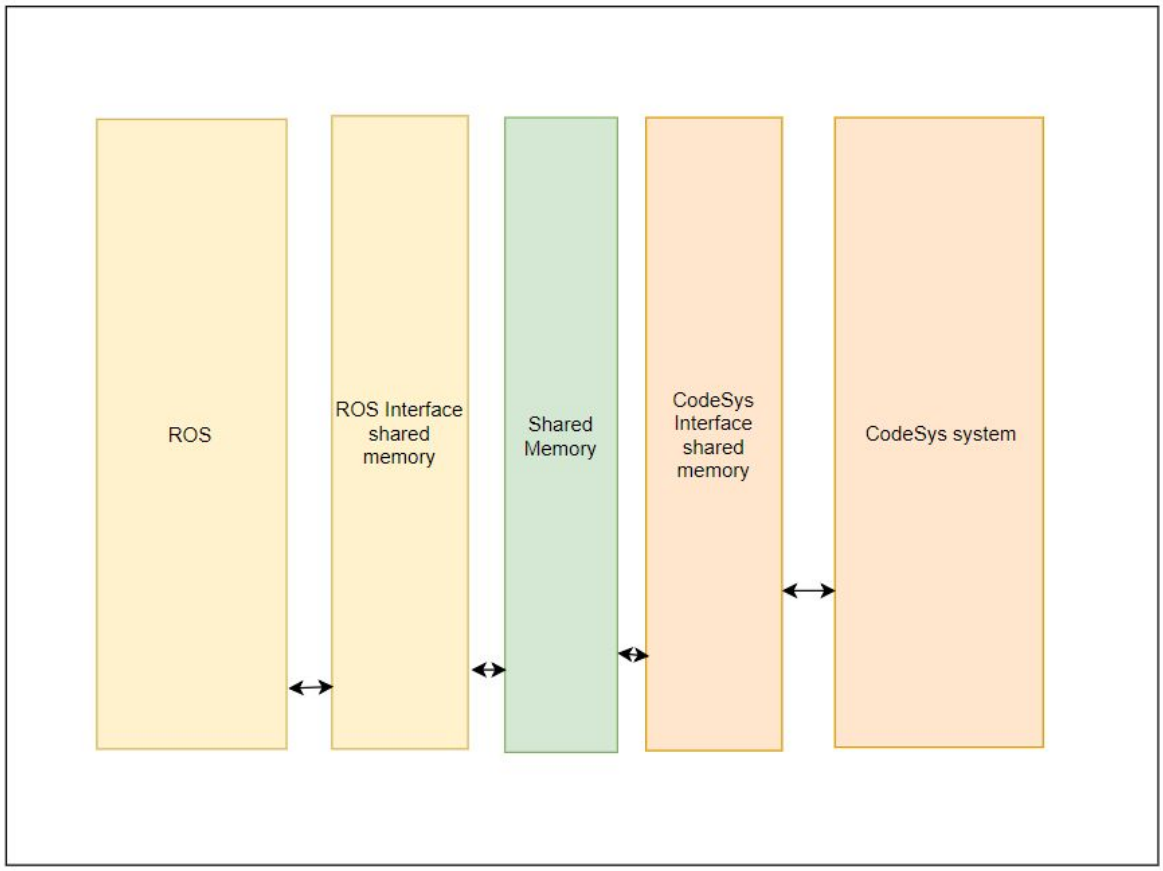

Figure 2: ROS and CodeSys on MH2

The first architecture is to run ROS and CodeSys on the PC-based motion controller (MH2). The communication between the two systems will be through the inter process mechanism (IPC) using shared memory.

The advantage of this architecture

1) Fast communication between ROS and CodeSys.
2) Implementing ROS solutions on a Delta Electronics' Technology.

The disadvantage of this architecture

1) ROS in windows is still experimental, no stable version release yet[2].

2) Add a mechanism to the shared memory to prevent data corruption. Cyclic Redundancy Check method as an example.
3) Windows can only run on two cores on MH2, the other two cores are reserved for CodeSys. Which means the performance is not very high.

2. ROS and CodeSys through shared memory.

ROS is a distributed computing environment, multiple nodes running on a multiple devices could communicate with each other through topics, that is one of the ROS advantage that we will be using in this architecture.

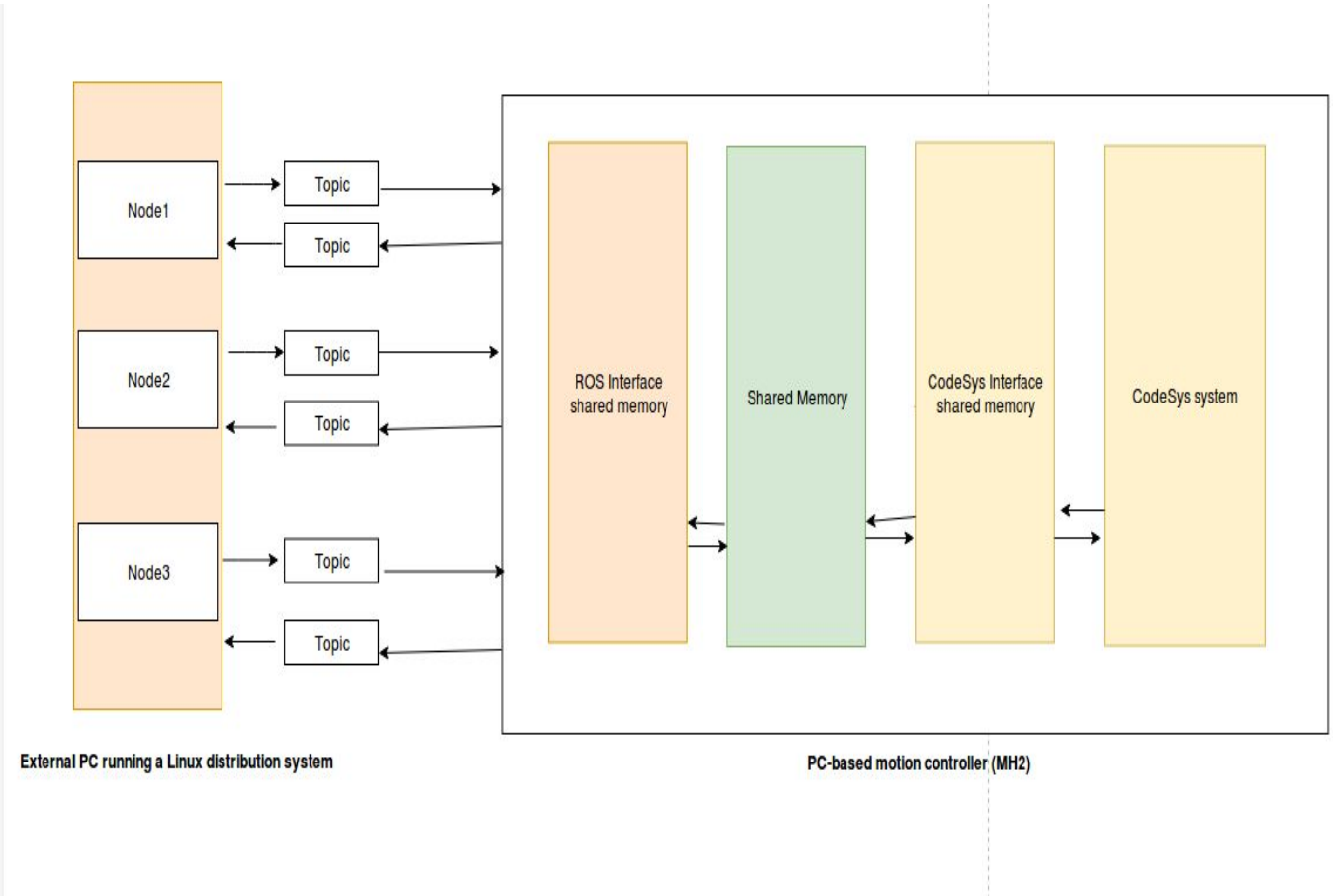

Figure 3: Shared Memory Architecture

As illustrated in figure 3, the ROS shared interface memory will be implemented on the PC-based motion controller and other nodes will be implemented on an external PC. The reason of this segregation is that there is only two cores free for the OS, thus the performance is reduced and also the ROS on windows IoT Enterprise is still experimental and there is not much support for it to implement the path planner.

ROS interface shared memory will subscribe to the desired topics, read their values and write them to the shared memory.
ROS interface shared memory will read what is written by the codesys interface shared memory and publish them to the right topics in the external device.

To avoid memory corruption and synchronize the read and write functions, the codesys library *SysSem* and *SysSemProcess* offer the synchronization process using semaphores.

In case the process 1 wants to read or write to the shared memory it will check if it is used by the process 2 or not. If it used by the process 2 the flag will be held up. When the flag is held down, the process 1 lock the semaphores and access the shared memory to manipulate the data.

The advantage of this architecture is

1) Fast communication
2) Implementing part of ROS solutions on Delta Electronics' Technology.

The disadvantage of this architecture is

1) Usage of two computers.
2) ROS in windows is still experimental, no stable version release yet.
3) Add a mechanism to the shared memory to prevent data corruption. Cyclic Redundancy Check method as an example.

3.ROS and Codesys communication through TCP/IP socket and shared memory

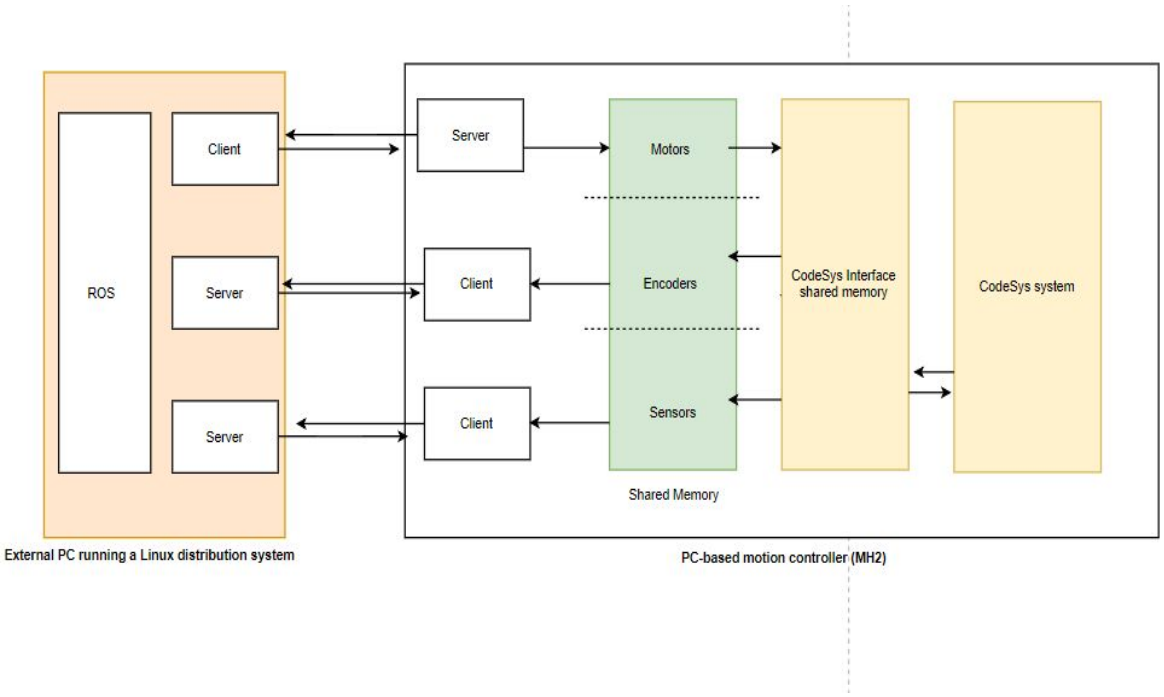

Figure 4: TCP/IP socket with shared memory architecture

As illustrated in figure 4, the communication between ROS and CodeSys could be done through the TCP/IP sockets. ROS will be implemented completely on an external device that is running a linux distribution and communicate to the CodeSys through the Client/Server mechanism.

The shared memory will be divided according to the number of data in interest to make sure that the communication will be only in one direction.

To avoid the concurrency of one server with multiple clients, multiple servers have being created to server each specific client.

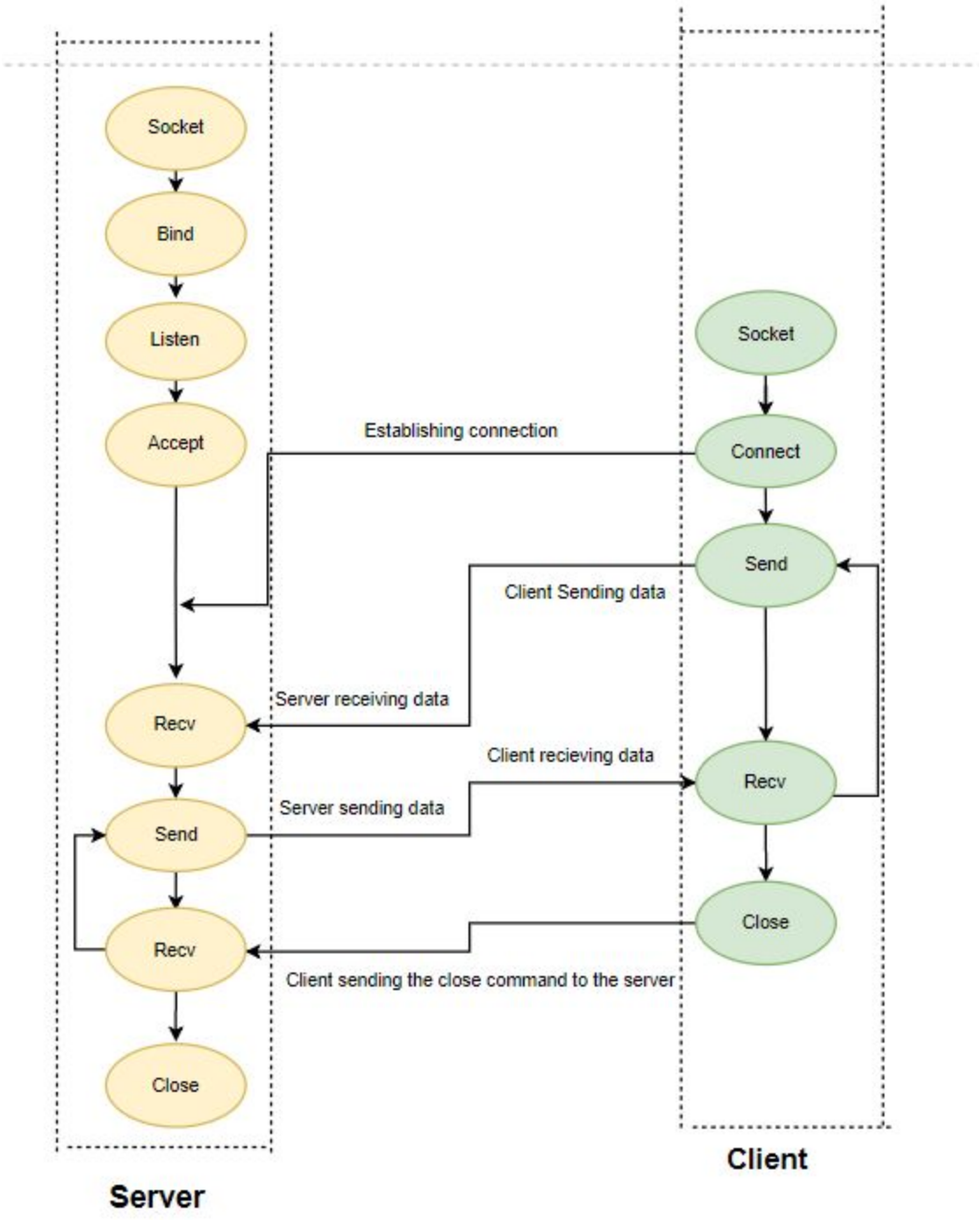

Figure 5: Client/Server flowchart

As illustrated in figure 5, the server is always listening to a certain network socket, when the client wants to establish a communication with the server he sends a request to that certain socket that the server is listening into.

The advantage of this architecture

1) ROS implemented completely on a linux distribution platform

The disadvantage of this architecture

1) Usage of two computers.
2) Add a mechanism to the shared memory to prevent data corruption. Cyclic Redundancy Check method as an example.
3) Slow comparing to the first architecture.

4. ROS and CodeSys communication through TCP/IP socket

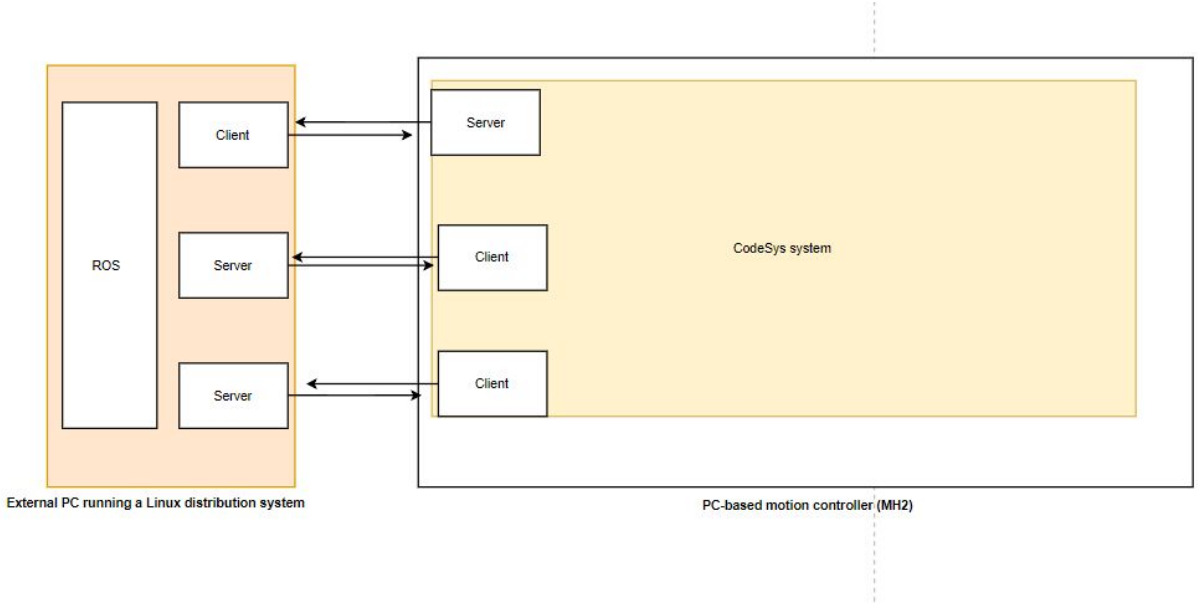

Figure 6: ROS and CodeSys communication

To avoid the latency added by the shared memory, the architecture illustrated in figure 6 establish a communication between ROS and CodeSys directly through TCP/IP socket without using the shared memory.

The advantage of this architecture

1) The validity of the data is ensured through the TCP protocol from the receiver and the sender.
2) ROS implemented completely on a linux distribution platform

The disadvantage of the architecture:

1) Usage of two computers.
2) Slow comparing to the IPC mechanism[3].

Conclusion

Since the Delta smart wrist is intended to be used in the industrial field, precision and security are a very high priority. ROS is still experimental on Windows, to avoid this problem ROS will be implemented on a Linux distribution device as shown in Figure 6 for the demo day.

ROS on windows will be investigated and tested more after the demo day, and may be used for the final demo.

Reference

[1]CODESYS Control RTE V3

[2]https://blogs.windows.com/windowsexperience/2018/09/28/bringing-the-power-of-windows-10-to-the-robot-operating-system/

[3]Embedded System Development with CODESYS and ROS Integration - Tiago Filipe Sousa Pinto