Communication between Tegra A and B

Python Test Program

The files *TegraCommunicationTestServer.py* and *TegraCommunicationTestClient.py* can be used to test the on-board switch (PEX8724) to make sure that it works before setting up a connection with ROS.

This is done by first confirming the internal IP addresses of each Tegra chip which can be done by running the **ifconfig** command in a terminal for reach Tegra chip. The internal IP addresses are given by the **eth0.200 inet** interface. The standard addresses should be **10.42.0.28** for Tegra A and **10.42.0.29** for Tegra B.

The variable *TCP_IP* is set to the IP address of the Tegra chip that is going to run the *TegraCommunicationTestServer* script. The variable TCP_PORT is set to 11311 in both scripts which is the standard port for the ROS master to test that it fully works.

After this is done the server script can be launched on the intended Tegra chip by running the command **python .\TegraCommunicationTestServer.py** and the client script on the other Tegra chip with the command **python .\TegraCommunicationTestServer.py**.

The output in the terminal on the client side should be as follows:

Hello Client 0
Hello Client 2
Hello Client 3
Hello Client 4
…

And the output in the terminal on the server should be as follows:

Hello Server 0
Hello Server 2
Hello Server 3
Hello Server 4
…

Where the number is increasing for each established connection to the server (once every second). If this is the case a successful connection has been established.

ROS Communication

To make the connection work with ROS the .bashrc file first have to be modified on both Tegra A and B. To open the file simply run **gedit .bashrc** from the user home directory in respective terminal. In the end of the file add the following lines:

ROS_IP=(THE LOCAL eth0.200 inet ADDRESS OF THE TEGRA)
ROS_HOSTNAME=(THE LOCAL eth0.200 inet ADDRESS OF THE TEGRA)
ROS_MASTER_URI=(THE URI TO THE TEGRA WHERE THE ROS MASTER IS EXECUTED)

An example for Tegra A where the ROS master is running on Tegra B can be seen below.

On the Tegra A machine add the following:

ROS_IP=10.42.0.28
ROS_HOSTNAME=10.42.0.28
ROS_MASTER_URI=http://10.42.0.29:11311

On the Tegra B machine add the following:

ROS_IP=10.42.0.29
ROS_HOSTNAME=10.42.0.29
ROS_MASTER_URI=http://10.42.0.29:11311

The IP addresses for respective Tegra is as stated in the previous section (Python Test Program). When the respective .bashrc file has been saved the terminals has to be restarted to load the correct information of the IP addresses. To make sure of this the following can be added in the .bashrc file.

echo "ROS_HOSTNAME: "$ROS_HOSTNAME
echo "ROS_IP: "$ROS_IP
echo "ROS_MASTER_URI: "$ROS_MASTER_URI

This prints the IP address for each parameter when the terminal is first opened.

After that can the connection be tested by simply running a ROS master and checking the topic lists available. Run the following command in the terminal on the machine intended to run the ROS master.

roscore

This will start the ROS master and should then be able to be reached from the other Tegra chip. This can be tested by running the following command.

rostopic list

This will display the available topics if the ROS master can be accessed and give an error message if it can't. If the connection is set up correctly it should be possible to run this command on both Tegras and see the same list.

If it works, the connection can be tested with pre-existing program *turtlesim* to see that it is possible to send ROS messages over the link. This is done by launching the turtlesim node on the Tegra chip that does not run the ROS master with the following command.

rosrun turtlesim turtlesim_node

If this works a graphical display with the turtle should appear. To control the turtle from the Tegra running the ROS master (i.e the tegra not running the turtlesim node) run the following command in a new terminal.

rosrun turtlesim turtle_teleop_key

This node makes it possible to control the turtle on the opposite Tegra chip with the arrow keys and should work if a connection has been established.

It should be noted that the key point to make the communication work is to have the correct IP corresponding to the Tegra chip running the ROS master. The nodes themselves can be divided according to preference.

Nvidia Drive PX2 Block Diagram

Below is a block diagram presented which illustrates the connection of all units on the board and how the Tegra chips are connected via the PEX9724 switch.