Connecting BeagleY-AI to an External Server

Last updated April 13, 2025

Purpose

This guide explains how to set up a TCP-based connection between BeagleBoard devices and an external Python server. It's aimed at enabling real-time multiplayer interaction, such as sharing the same image across multiple BeagleBoard LCD screens.

The best way to connect two devices in real-time is to use an external server with TCP constant connection. This allows the server to assign roles to different connections. You can set one as host and one as client. This is much harder to do via UDP connections.

The first step is to use the sockets library in C and Python use STREAM not DGRAM, this will enable TCP.
In the C sockets library, use the connect function to connect to the server IP and Port. In Python, import the sockets library as well. You should use the recv and sendall functions for TCP in Python to receive and send data back to the devices. Always make sure to encode this data to bytes before sending using the encode function. You should also decode data you receive using the decode function.

The second step is to complete port forwarding of the chosen port. I recommend port numbers higher than 3000 to be safe, but you can search up the table to ensure your port isn't already being used by a system service. Run "ip -a" on Linux and find your local ip for your server machine. Simply log into your home router and find the option to port forward. Enable TCP, input your local IP and set your port. Apply this and test if your port is open by pinging your external IP at the chosen port.

Next, you can accomplish host client or other roles using simple booleans or ints. When the first client connects, assign it a value on the outer scope of the thread loop so that the value is still accessible inside the loop. Also, set a value in the server object that indicates a role has been consumed by one device. This can be an int or a bool, such as setting self.hasHost to true. Therefore, when the next device connects, simply check this server value and assign a new role accordingly.

The best way to organize your server code is to make a Server python class and run the object via a main.py file by importing the server. Then this can simply be run by running python3 main.py in the directory.