Python Server: This code implements a Python server that listens for incoming connections from Python clients, receives data from them, saves the data to the appropriate directory depending on the file prefix, and sends the data to a Unity client.

The server is defined as a class **SensorServer**, and it takes three arguments: **host**, **port**, **hub1\_dir**, and **hub2\_dir**. **host** and **port** define the address and port number on which the server will listen for incoming connections. **hub1\_dir** and **hub2\_dir** are the paths to the directories where the data received from Hub1 and Hub2 will be saved, respectively.

The server listens for incoming connections in an infinite loop using the **listen()** method. When a new connection is established, a new thread is created to handle the client using the **handle\_client()** method. The **handle\_client()** method receives the file name, file data, and file prefix from the client and saves the file to the appropriate directory. It then sends the file data back to the client using the **conn.send()** method.

The **broadcast()** method sends the file data to all connected clients. However, this method is currently commented out in the code.

Overall, the code appears to be functional and handles incoming connections and data from clients appropriately.

Top of Form

Bottom of Form

Unity Client: This script defines a Unity client that receives CSV files from a server and saves them to the appropriate folder based on the file name. The client connects to the server with a given IP address and port. The server sends CSV files to the client, and the client saves the files in either the "Hub1\_data" or "Hub2\_data" folder depending on the file name prefix.

Here is an overview of the script:

1. Set the server's IP address and port, and the time interval for sending/receiving data in seconds.
2. Define the path for "Hub1\_data" and "Hub2\_data" folders in the **Awake()** method and create those directories if they don't exist.
3. In the **Start()** method, create a new thread that runs the **ClientThread()** method for connecting to the server and receiving data.
4. In the **OnApplicationQuit()** method, close the client socket and stop the client thread.
5. The **ClientThread()** method: a. Connects to the server using the given IP address and port, creating a new TCP client socket. b. Reads the filename length from the server, then reads the filename itself. c. Filters out illegal characters in the filename and extracts the relevant part of the filename containing the ".csv" extension. d. Reads the data length from the server, followed by the actual file data. e. Determines the appropriate output directory based on the file prefix, and then saves the received file in that directory. f. Logs a message stating that the data has been received and saved. g. If any exceptions occur, they are logged, but the program does not terminate.