Week 07: TCP Communication in C++/CLR

What is TCP communication

- Communication is often required between different devices (computer to computer or computer to peripheral).

- If these devices are connected via a network, specific internet protocols are required to communicate.

- One of the most common communication protocols is TCP (transmission control protocol).

What is TCP communication (cont.)

- TCP provides a reliable means of data transfer as it is designed such that there is a ‘handshake’ between communicating devices and confirmation that all the data was received correctly.

- When communicating, the device receiving requests is called the server (this will the UGV for your assignment) and the device sending the requests is the client (this will be your program).

What do you need as a client

When communicating your program needs to do a few things:

- Establish a connection (this will open a link between the two devices)

- Send data packets (requests)
- Receive data packets (responses)

- Close the connection (confirm that all communication has completed)

Establishing a connection

```
// Create a client object that will hold the connection to the server
TcpClient^ Client = gcnew TcpClient("x.x.x.x", PortNumber);
// Client properties can be set as required (see
https://learn.microsoft.com/en-
us/dotnet/api/system.net.sockets.tcpclient?view=net-6.0)

// Get a network stream object for writing and reading
NetworkStream^ Stream = Client->GetStream();
```

Sending data packets

```
// When forming data packets, they may be encoded in different formats
// A common way is ASCII communication as shown below
array<unsigned char>^ SendData = gcnew array<unsigned char>(16); // Initialise variable to hold string that will be sent
SendData = System::Text::Encoding::ASCII->GetBytes("send string"); // Convert the given string to a buffer that can be sent

// Data can be sent as a whole using
Stream->Write(SendData, 0, SendData->Length); // Send the data in the SendData buffer, starting at the 0th byte, and specifying the buffer size

// A single byte of data can be sent using
Stream->WriteByte(0x03); // Send the specified hexadecimal byte
```

Receiving data packets

```
// Read the incoming data from the TCP stream

array<unsigned char>^ ReadData = gcnew array<unsigned char>(16); //
Initialise variable to hold response

Stream->Read(ReadData, 0, ReadData->Length); // Read the response into the
buffer ReadData starting at the 0th byte and having the specified size


// Convert the data given back into ASCII for processing

String^ ResponseData = System::Text::Encoding::ASCII->GetString(ReadData);


// Another useful member variable (*not a function) that checks if data is
currently ready to be read

Stream->dataAvailable;
```

Closing the connection

```
// Close the stream and client
Stream->Close();
Client->Close();
```