This documents serves as supplemental requirements for the course project. Please abide by these standards to ensure cross-compatibility with the code implementations of your peers.

Setup and Specifications

From Client to Middleware

The client serves to issue requests to the robot by communicating through the middleware. The client additionally receives and documents the responses from the robot, also communicated through the middleware.

To run the client, the following command-line parameters must be used:

executable_name IP/host_name port ID L N

- executable_name - the decided name for the compiled code; for consistency, let's use **robotClient**
- IP/host_name - the dotted decimal IP address or host name of the server; your code must support host name resolution if needed
- *port* - the port that the server is running on
- *ID* – the ID of the robot that the client wishes to communicate with; this will serve as a password to be verified by the middleware
- L - the length of each side of the polygonal travel path
- N - the shape of the polygonal travel path

Given a particular *L* and *N*, the client will individually communicate the commands necessary for the robot to complete its travel, waiting for each command to finish before sending the next. Command messages have the following format:

0 31

Client-selected communication ID (unsigned integer)
… Robot ID (NULL-terminated string) …
… Robot Command (NULL-terminated string) …

Note that we assume this robot command to fit within a single UDP message. The commands are as follows:

To get an image of the robot's current perspective: **GET IMAGE**

To get the robot's current GPS location: **GET GPS**

To get the robot's current differential GPS location: **GET DGPS**

To get the robot's current laser data: **GET LASERS**

To move the robot forward: **MOVE [velocity]**

To rotate the robot: **TURN [velocity]**

To stop the robot: **STOP**

From Middleware to Robot

The middleware interprets the client's requested command and packages into an appropriately formatted HTTP GET request – see the details of Project #3 for proper HTTP format.

To run the server, the following command-line parameters must be used:

executable_name server_port IP/host_name ID image_ID

- executable_name - the decided name for the compiled code; for consistency, let's use **robotServer**
- server_port – the port that the server will run on
- IP/host_name - the dotted decimal IP address or host name of the robot; your code must support host name resolution if needed
- *ID* - the ID that the robot corresponds to; this will be used to communicate with the robot and to verify the ID sent by the client
- *image_ID* – the ID that will be used to ascertain a snapshot from the robot

The robot's port is not specified in the command-line because each command corresponds to a port:

Port 8081: **GET IMAGE**

Port 8082: **GET GPS**, **MOVE [distance]**, **TURN [degrees]**, **STOP**

Port 8083: **GET LASERS**

Port 8084: **GET DGPS**

From Middleware back to Client

Sending the data from the middleware back to the client will require some overhead to ensure some reliability. Each UDP message from the middleware to the client will be formatted as follows:

0 31

Client selected ID number
Number of UDP Messages to Send
UDP Message Index (Starting at 0)
…   Segment of Robot's Response   …

It is the responsibility of the client to reorganize the UDP messages in the event that they arrive out of order.

Constants

For consistency, there are two major constants that are required to use across all implementations:

The size of a single UDP message (including overhead): **1000 bytes**
The client timeout delay before printing an error: **5 seconds**

Closing Comments

Any other implementation design choices are entirely up to your team. This document is only meant to standardize inter-program communication and is not meant to restrict the creativity of your implementation.