BTN415 Term Project, Winter 2017

Robotic Communication

In this term project, working in teams of two or three, you will:

- Review a pre-defined application layer protocol
- Design a multi-threaded client application using a reliable TCP/IP communications link
- Encapsulate the application layer protocol inside the TCP/IP client application to command and communicate with a mobile robotic device
- Use your client application to control the robot and receive its telemetry sensor data in a demonstration exercise

LEARNING OUTCOMES

Upon successful completion of this term project, you will have demonstrated the ability to:

- Interpret an application layer protocol
- Implement a protocol definition within a client environment
- Designed a multi-threaded application
- Encapsulate an application protocol with TCP/IP reliable communications

OVERVIEW

In this term project your team will design and develop a client application that will simulate the operations station for a tele-operated robot. A mobile robot has been designed and built using a unique application layer protocol. The application layer protocol is defined in the TermProject_Milestone1 document.

The robot runs as a **Server** with two TCP socket connections. The first socket connection is used for the robot to send a telemetry stream of sensor dat. The second socket connection is used for the robot to receive commands from the user . The robot itself is designed and operating as a Wifi access point. Once the robot is operational (*acting as a server*) it will open the two sockets described before, and put them in listening mode for your client software to connect with.

After the TCP Handshaking has completed on the telemetry socket the robot will start streaming data packets at a rate of 1 packet every 5 seconds. These data packets will contain telemetry information from the Sonar and Potentiometer sensors onboard.

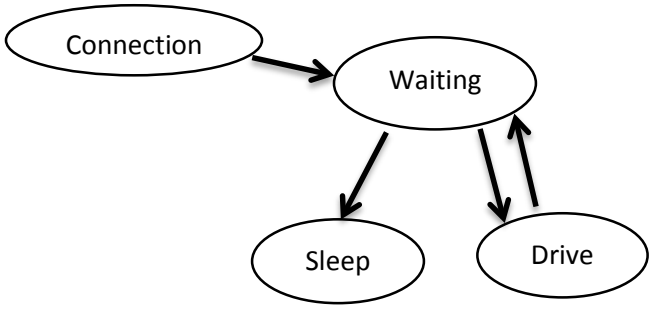
After the TCP Handshaking has completed on the command socket the robot will enter a "Waiting" mode.

Waiting – In this mode the Robot waits for a command packet from the operations software.

Drive – The robot transitions to "Drive" mode when it receives a drive command from your client application. Once the packet is validated, the robot will respond with an ACK packet and then execute the command. Upon finishing the command the robot returns to the "Waiting" mode and waits for the next command.

Sleep – The robot transitions to "Sleep" mode when it receives a sleep command. Once the packet is validated, the robot will respond with an ACK packet and then cease all communications with your client application until a hard reset is performed.

The following state diagram gives a visual representation of these modes:



The demonstration aspect of this term project is a competition. You will have to use your client software to control the robot and perform a set of tasks. Your execution times, accuracy and procedures for executing the tasks will be judged against the other teams.

NOTE
To make things interesting…. You will not be allowed to connect to the robot until your competition day!

(don't panic – Sim Provided)

This project will be tracked and broken down into the following milestones (deliverables). For due dates and timelines, please speak with your professor.

- Milestone 1: Your team have to design, implement and test a class to handle the application layer protocol. Objects of this class will be used to create packets to be sent/received to/from the robot. Details about the application protocol and required deliverable items can be found in the Milestone 1 specifications document. A test **main.cpp** file will be provided.

- Milestone 2: Your team have to design, implement and test a class to handle socket communications. Objects from this class will be used to send command packets and receive telemetry packets during operation of the robot.
- Milestone 3: Using your newly developed classes, create a multi-threaded application that will open TCP/IP communications with both the Command and Telemetry sockets on the robot. You will be provided a binary executable client to test your connections with. Details about the communications link and instructions for running the test client can be found in the Milestone 2 specifications document.
- Milestone 4: Competition Day. You will connect your client software to the physical robot and command the robot to execute the task(s). Details about the set of tasks to be performed will be provided closer to the competition date.

Setting Up Your Groups

This project is a group project of 2-3 students. You are **NOT** allowed to work alone. Once you have selected your groups send an email to your professor with the names and student numbers of each of your group members. Your professor will setup the group on Blackboard for submitting your milestones.

SUBMISSION REQUIREMENTS

One person from each group will submit the group's work to Blackboard. For each deliverable milestone (listed above) you will submit the following:

1. Milestone 1:
    - Project Source Code – This is a single ZIP file containing the following files:
        - Your Visual Studio project files
            - ***Delete all your debug and release directories***
            - ***Delete all database files***
        - README.txt file containing any execution instructions (if required)
        - Any input files required (test inputs, etc…, if required)
2. Milestone 2:
    - Project Source Code – This is a single ZIP file containing the following files:
        - Your Visual Studio project files
            - ***Delete all your debug and release directories***
            - ***Delete all database files***
        - README.txt file containing any execution instructions (if required)
        - Any input files required (test inputs, etc…, if required)

3. Milestone 3:
    - Project Source Code – This is a single ZIP file containing the following files:
        - Your Visual Studio project files
            - ***Delete all your debug and release directories***
            - ***Delete all database files***
        - README.txt file containing any execution instructions (if required)
        - Any input files required (test inputs, etc….)
    - In class competition. No deliverables to be submitted.