BTN415 Term Project, Winter 2015

Robotic Communication

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

- Review a pre-defined application layer protocol
- Design a Server application using a reliable TCP/IP communications link
- Encapsulate the application protocol inside TCP/IP to command and communicate with a mobile robotic device.

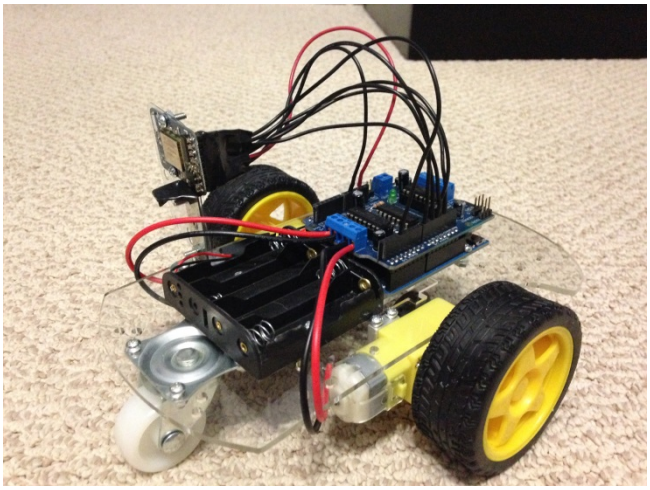
LEARNING OUTCOMES

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

- Interpret an application protocol
- Implement a protocol definition within a Server environment
- Encapsulate an application protocol with TCP/IP reliable communications
- Write a technical paper on data communications

OVERVIEW

In this term project your team will design, develop and document a server application that will simulate the ground station for a space exploration robot. A mobile robot has been designed and built using a unique application layer protocol (defined below).



The robot runs as a client and has four modes of operation; Waiting, Drive, Status and Sleep. The robot starts in Waiting mode after being turned on.

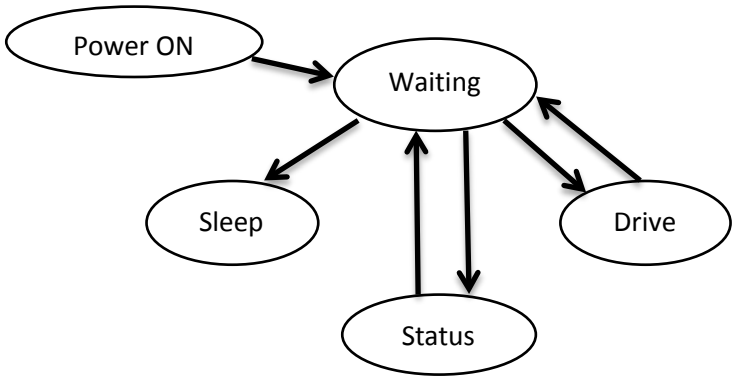
Waiting – In this mode the Robot (acting as a client) will connect to the Wifi access point and request a connection with the Server (your team's software)

Drive – The drive mode works autonomously from the server. The robot receives a set of drive commands when connected to the server, validates the command packet, disconnects and execute the command sequence. Once completed the robot goes back to Waiting mode, reconnects to the server and waits for the next set of commands. This process keeps looping until the robot is put to sleep.

Status – The status mode is operated with a connection to the server. When the server sends a status command to the robot, it will respond immediately with a sequence of values, as defined by the application layer protocol below. Following, it it will return to Waiting mode

Sleep – In this mode the robot will cease all communications with the server 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 BTN415 (all sections) competition. You will have to use your server software to control the robot and perform a set of tasks. Your execution times and procedures for executing the tasks will be judged against the other teams of BTN415.

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

(don't panic – keep reading)

The execution and tracking of this project will be done through Blackboard and Groups. Each group will be responsible for uploading the deliverables to Portfolios.

PROCEDURES

Application Layer Protocol Definition

COMMANDS

The application layer protocol contains three main components. A header which contains a **PacketID** and **CmdListSize**, a body with the **DriveCmdList,** and a trailer with a **Parity** validation. Definitions of these elements are as follows:

- **PacketID** – contains the unique command type (**DRIVE, STATUS, SLEEP**)
- **CmdListSize** – contains the number of **DriveCmds** that have been listed in the body of the packet
- **DriveCmdList** – the drive command list contains 1 to 10 drive commands. Each drive command is defined as follows:
  - **DriveCmd**
    - **Direction** – the drive command directive value
    - **Duration** – the number of seconds to execute the direction directive
- **Parity** – the packet validation value to ensure correct transmission

The following is a visual representation of the application layer protocol for the mobile robot:

Drive Command:

Packet Header		Packet Body	Packet Trailer
PacketID	**CmdListSize**	**DriveCmdList**	**Parity**
1-byte	1-byte	Max 80-bytes (0-10 DriveCmds)	1-byte

DriveCmd	
Direction	**Duration**
4-bytes	4-bytes

Status/Sleep Commands:

Packet Header		Packet Trailer
PacketID	**CmdListSize**	**Parity**
1-byte	1-byte	1-byte

Drive Command and Packet Type Values

The pre-defined command and packet types are defined as follows:

- Drive Command Directives
  - FORWARD 1
  - BACKWARD 2
  - LEFT 3
  - RIGHT 4
- Packet ID Types
  - DRIVE 15
  - STATUS 5
  - SLEEP 0

RESPONSES

The response message from the mobile robot will use the same header and trailer as the command (I.E. The PacketID = STATUS and the Parity will be calculated using the same algorithm as above). The body of the message will be populated with 10-bytes of information.

The following is a visual representation of a response packet:

Packet Header		Packet Body	Packet Trailer
PacketID	**CmdListSize**	**StatusData**	**Parity**
1-byte	1-byte	10-bytes	1-byte

Parity Algorithm & Example

The parity check performed by the robot is a simple count on the number of **BITS** set to '1'. For example:

Header		Body		Trailer
PktID 2	CmdListSize 1	Direction 1	Duration 15	Parity
00000010	00000001	0000000000000001	0000000000001111	
00000010000000010000000000000001000000000001111				7

Server Requirements

Your team is responsible for designing and implementing a Server application that simulates the ground station the robot will communicate with. Team member tasks should be divided as follows:

- Defining and coding the packet protocol
- Writing the member functions for generating Drive Command Packet
- Writing the member functions for generating Status and Sleep Command Packets
  - Also the functionality for processing and displaying the received status response
- Writing the member functions for handling the TCP/IP communications

Your source code should be written with the following modules:

- **SERVER.CPP** – Contains the **main** and performs the following functionality:
  - TCP/IP connection control
  - Collects user input (commands for the robot)
  - Sends/Receives packet data between the Server and mobile robot
  - Displays any status/error messages generated or received from the mobile robot
- **PKT_DEF.H** – Contains the **class PktDef** definition for creating, managing and decoding the application layer protocol.
- **PKT_DEF.CPP** – Contains the implementation of all the member functions defined in **PKT_DEF.H**. Functionality should include:
  - Memory Management
  - Generation of command packets (headers, body and parity checks)
  - Processing response packets (parity checks, extracting data)

NOTE:

There are no GUI requirements. You are free to build the user interface however you feel fit.

Technical Paper

Part of this assignment is to give you experience writing a technical paper. This paper will act as a design document and user manual for your server. *Using the template provided on blackboard* write a short technical paper (3-4 pages in length) that documents your server design. Information that should be included in this document is as follows:

- Introduction. Describe (high level) what your Server does and how it works.
- Details about your Server design. How did you organize your code, classes/structures implemented and how a user interacts with your Server software to control the mobile robot.
- The protocol used. How did you implement the protocol above and encapsulate it in your TCP/IP reliable connection. Provide pictures and diagrams to help describe your design.
- Summary. Provide details about how you tested your Server software and issues you ran into and solved along the way.
- References. You should list all references used during the design and development of this term project. Code reused, websites that help you debug problems, etc...

Simulation Software

In lieu of being allowed to test using the mobile robot, a piece of simulation software has been developed. This simulation software is written for a Windows PC and runs as a console application. It simulates the exact behavior of the software running on the mobile robot. Follow the instructions below to run the simulator with your code:

1. Start running your Server application
   a. HINT: It should open a listening port for the mobile robot to make a connection with.
2. From the windows command prompt run ***StudentSimulator.exe***

   a. The ***StudentSimulator.exe*** will connect to port 5000 on the local host (127.0.0.1)
3. Enter user information on your Server GUI and transmit the data to the ***StudentSimulator.exe***
4. The ***StudentSimulator.exe*** will receive your command, validate and execute exactly the same way the mobile robot will. The only extra feature is the ***StudentSimulator.exe*** will provide you with limited debug information if failures have occurred. The following is an example:

```
DRIVE Command with 1 commands has been received
      **** ERROR:  Invalid Drive command has been received

********Suggestions***************
      Check your packet body definition
      Check your command definition
      Check your packet generation function
```

Competition

The demonstration will be a competition between the project groups across both sections of BTN415. There will be 4 possible time slots for you to demonstrate/compete with your server application. Signup sheets will be posted in the ICT-Office. See Blackboard for details on the location of the signup sheet.

During this competition, you will be required to use your Server application to command the BTN415 robot in a figure-8 around two desk chairs. This sounds easy, but a word of caution. This is not a multi-million dollar NASA robot. It does not drive straight and it's not accurate at turning.

Your team will be evaluated based on the following criteria:

- The time to complete a figure-8 (max time allotted is 20 minutes)
- Were any objects hit during motion
- Drive strategy. i.e. the use of single commands or multiple commands
- Request for Status
- Command to Sleep

Major Milestones/Deliverables

Your team has 4 weeks to complete the project, but there are three deliverables that are due throughout the 4 weeks of work. Details about the dates/times of these deliverables will be posted on Blackboard and discussed by your professor, but based on the semester schedule the project timelines are defined as follows:

Deliverable Number	Description	Week Due
N/A	Term Project Released to Students	March 27
1	Competition/Demonstrations	April 10
2	Final source code and ZIP project files (as per the submission below)	April 11

4	Technical Paper – Your groups design for the Server	April 17

Submission requirements for each of these deliverables can be found in the next section.

SUBMISSION REQUIREMENTS

Each group contains a Portfolio for submitting documents. For each deliverable milestone (listed above) you should submit the contents to a journal posting in your groups portfolio. In order for your term project to be completed you should have the following posted:

1. Server Source Code & Design – This is a single ZIP file containing the following files:
   - Your Visual Studio project files
     - ***Delete all your debug and release directories***
   - README.txt file containing any execution instructions (if required)
   - Any input files required (test inputs, etc….)
2. A technical paper describing the design of your server application
     - ***Document must be formatted using the template provided***
3. A file summarizing to your instructor which group members worked on which parts of the term project.