HRI Lab #1 SP2020, Part 1, 2024-Jan-22

In this lab you will use an executable that takes commands from UDP and forwards them to the robot simulator for viewing and potential forwarding to the physical robot. In Part 1.1, you will first explore the robot's movement space in the Actin Viewer. In Part 1.2, you will set up a connection to the Actin Viewer and run an executable that will forward commands sent over UDP. In Parts 1.3 and 1.4, you will use modules from MiniVIE to send UDP commands to the two different Cyton robot simulation environments. In Part 1.5, with any left over time, you will explore how to move the Cyton robot to some desired positions.

In Parts 1.3 and 1.4, you will be accessing functions from the MiniVIE code project. These functions are referenced as "Static Methods" which means that they are functions contained within user defined classes. Those not familiar with Object Oriented Programming in MATLAB should refer to documentation under MATLAB -> Advanced Software Development -> Object-Oriented Programming.

References:

Cyton Viewer: http://outgoing.energid.info/documentation/cyton/db/d53/PAGE_CytonViewerIntro.html Remote Command API: http://outgoing.energid.info/documentation/cyton/db/d2f/PAGE_RCApi.html

Part 1.1: Control the virtual robot

Actin Viewer Software is the primary interface to both the physical and the virtual robot (since this contains all safety limits and calibrations for the physical robot.

1. Start Actin Viewer on your computer. Once started, hit "play" on the toolbar of the simulation.
2. Try moving the robot by controlling the joints directly in the GUI viewer. If the joint sliders are not visible, click the button on the toolbar that is two to the left of the "E-Stop button".
3. Also try moving the robot by dragging the directional arrows around the gripper. To enable this mode, you may need to click on the toolbar button two to the right of the stop simulation button. You may also need to place the robot in a position that is not at the end of its joint boundaries using the individual joint sliders.
4. Note how each individual joint must move to generate smooth movements of the end of the robot--for example, do left right movements mostly use one joint, or more?

Part 1.2: Connect to the virtual robot

Third-party command modules within the Actin Viewer Software (joysticks, network, etc. are controlled via 'plugins' to the Actin viewer software.

Within the Cyton Actin Viewer, select:

- Plugins
- Load Plugin
- remoteCommandServerPlugin.ecp

Then press Play on the simulation.

The robot can now accept third party commands. Within this folder, find and run cytonCommandExample.exe to initialize. The Cyton robot should first go home, then to a manipulation position.

Part 1.3: Control the virtual robot in Cyton Viewer

To simplify interfaces between the robot and the human operator, we will be using a custom "MiniVIE" code project for Myoelectric and neural signal processing and pattern recognition: https://www.bitbucket.org/rarmiger/minivie/

This code project uses MATLAB

```
% Setup the MiniVIE environment and return to your current directory
currDir = cd;
cd('C:\GitHub\MiniVIE');
MiniVIE.configurePath();
```

```
[MiniVIE.m] Configured MiniVIE path at: C:\GitHub\MiniVIE
```

```
cd(currDir);

% A udp network connection can be established to send joint
% commands to port 8888.  The format is an array of 8 float64 values. The
% MiniVIE\Utilities\PnetClass.m makes sending these messages easy

% initialize the MATLAB UDP object
udp = PnetClass(8889, 8888, '127.0.0.1');
udp.initialize();
```

```
[PnetClass] Opened pnet socket #39 at local port: 8889; Default destination: port 8888 @ 127.0.0.1
```

```
% set the current angles of the robot
currentAngles = zeros(1,8);

% generate some new angles for the robot to go to
q = [0 -0.7 0 -0.7 0 -0.7 0];

% create a vector that includes the robot's arm angles and the gripper
% distance
desiredAngles = [q, 0.00003];

% send the data to the Actin Viewer
udp.putData(typecast(desiredAngles,'uint8'))

% get the robot's current joint angles from the robot's sensors
bytes = udp.getData()
```

```
bytes =

     []
```

```
angles = typecast(bytes,'double')
```

```
angles =

     []
```

```

% get documentation on PnetClass from MiniVIE
help PnetClass
```

```
  PnetClass Class for interfacing pnet tcp-ip-udp library
     Wrapper for pnet udp functions

   Handles some of the non-intuitive cases with pnet to prevent timeouts
   and also to read down a fast filling buffer without delay

   Usage:
     obj = PnetClass(localPort,remotePort,remoteIP);         % create object
     [success, msg] = initialize(obj);                       % open local port
     [dataBytes, numReads] = getData(obj);                   % read down buffer and return only the latest packet

   27-Feb-2013 Armiger: Created

     Documentation for PnetClass
```

Part 1.4: Control the virtual robot in Unity

For this part, open the vCyton application in the hrilabs/Applications/vCyton folder.

A udp network connection can be established to send joint commands to port 12001 and receive joint angles on port 12002. The format is an array of 8 float32 values (singles in MATLAB).

```
% initialize the MATLAB UDP object
udp = PnetClass(12002, 12001, '127.0.0.1');
udp.initialize();
```

```
[PnetClass] Opened pnet socket #12 at local port: 12002; Default destination: port 12001 @ 127.0.0.1
```

```

% set the current angles of the robot
currentAngles = zeros(1,8);

% generate some new angles for the robot to go to
%q = [0 0.7 0 0.7 0 0.7 0];
q = [-2.617901 1.91986 0 0.210654 1.878615 0.513543 -0.376678];

% create a vector that includes the robot's arm angles and the gripper
% distance, and convert to a float/single
%desiredAngles = single([rad2deg(q), 0.01]);
desiredAngles = single([rad2deg(q), 0.001]);

% send the data to the Unity application
udp.putData(typecast(desiredAngles,'uint8'))

% get the robot's current joint angles from the robot's sensors
```

```
bytes = udp.getData();
angles = typecast(bytes,'single')
```

```
angles =

  0×0 empty single matrix
```

Part 1.5: Explore

Using the UDP connection:

1. Find 2 example positions where the robot contacts the table in the Actin Viewer simulation

q = [1.006841 1.356764 -1.102421 0.776194 -1.143668 0.513543 -0.376678];

angles = 0.0067   0.3542   0.4980

q = [-2.617901 1.91986 0 0.210654 1.878615 0.513543 -0.376678];

angles = -0.5474   0.1526   0.1019

1. Find the values that represent commands for full opening and full closing of the gripper

0.015 for fully open

0.00003 for fully closed

Submitting This Lab

Export this *.mlx file as Lab1a_##_<LastName1>_<LastName2>.pdf (## should be your computer number) and email to Lauren.Diaz@jhuapl.edu