UNSW SYDNEY

Real Time Data Exchange (RDTE) Toolbox

Summary

- Provides a scripting interface to allow for easier manipulation of the UR5e
- Sends Urscript commands as byte strings to the UR5e
- Can send:
  - Movej
  - Movel
  - Movec
  - Movep
- Can receive:
  - Pose/Joint position
  - Joint Velocity
  - Joint Acceleration
  - Joint Torque

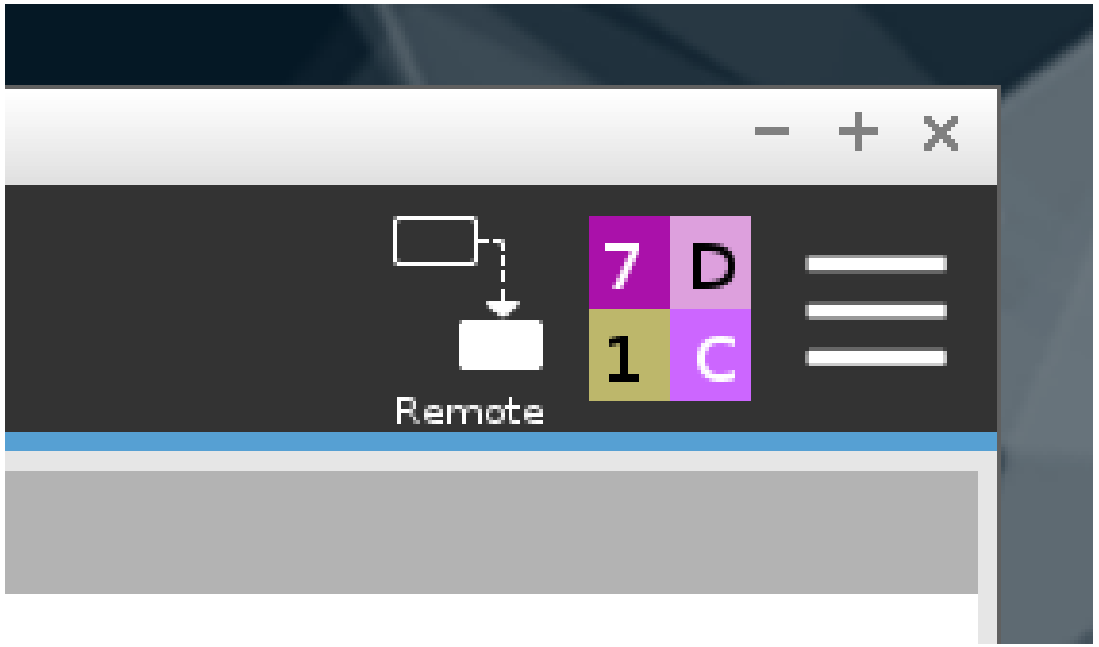
Setup

1. VM Installation
   1. Virtualbox : Almost everyone
   2. VMware for Mac with M1’s : Only if you can’t install VirtualBox
2. Setup port forwarding (Only required for Virtualbox. If you are using VMware don't worry about this)
3. Setup RTDE folder path in MATLAB

Laptop Side setup: Everyone

1. Connect ur5e ethernet cable to laptop or pc (Orange cable)
2. Go to ethernet settings
3. Change to manual
4. Turn on IPv4
5. Set IP address to “192.168.0.77”
6. Set subnet mask to “255.255.255.0”

For operation



- Make sure that the UR5e is In remote control mode.
  - Switch to automatic then to remote control mode

This is only required on the actual UR5e. Not needed on the VM

TCP Connection Setup

Only once in the program

- Port = 30003
- When connecting to the VM: Only for Virtualbox. For Vmware users, it is different. Please talk to a demo if you don't know what the IP address is.
  - Host = 127.0.0.1
- When connecting to the UR5e:
  - Host = 192.168.0.100
- Finally call the constructor:
  - rtde = rtde(host,port)

- Host on the UR5e must be configured correctly for this to work. The demonstrators will have already set this up.
- You can also use your own laptop by connecting the ethernet port directly to your laptop. You will also need to configure the ethernet connection on your PC for it to work.

Functions

movej : Move to position (linear in joint space)

When using this command, the robot must be at a standstill or come from a movej or movel with a blend. The speed and acceleration parameters control the trapezoid speed profile of the move. Alternatively, the t parameter can be used to set the time for this move. Time setting has priority over speed and acceleration settings

Definition:

```
[poses,joints,jointVelocities,jointAccelerations,torques] = movej(target,jointOrPose,a,v,t,r)
```

movej : Move to position (linear in joint space)

[poses,joints,jointVelocities,jointAccelerations,torques] = movej(target,jointOrPose,a,v,t,r)

PARAMETERS:

- **Target =** Joint position or pose.
- **jointOrPose** = "joint" or "pose" to provide joint inputs or pose inputs
- **a** = joint acceleration (m/s^2) (default = 1.4)
- **v** = joint speed (m/s) (default = 1.05)
- **t =** time (s) (default = 0). If specified it will ignore the **a** and **v** values
- **r** = blend radius (of target pose) (m) (default = 0)

movej : example usage

target = [-588.53, -300, 200, 2.2214, -2.2214, 0.00];

	command
Output pose and input xyzrpy	pose = rtde.movej(target)
Output pose and joint position and input xyzrpy	[pose,jointPos] = rtde.movej(target)
Output pose, joint position, joint velocity	[pose,jointPos,jointVel] = rtde.movej(target)
Output pose, joint position, joint velocity, joint acceleration	[pose,jointPos,jointVel, jointAcc] = rtde.movej(target)
Output pose, joint position, joint velocity, joint acceleration, joint torque	[pose,jointPos,jointVel, jointAcc, jointTor] = rtde.movej(target)
Only want joint position	[~,jointPos] = rtde.movej(target)
Only want joint velocity	[~,jointPos,~] = rtde.movej(target)
Want joint position and joint acceleration	[~,jointPos,~, jointAcc] = rtde.movej(target)

movej : example usage

target = [-2.4670 -2.1287 -1.6192 -0.9645 1.5708 -2.4670];

	command
Output pose and input joint position	pose = rtde.movej(target,'joint')
Output pose, joint position, joint velocity, joint acceleration, joint torque and input joint positions	[pose,jointPos,jointVel, jointAcc, jointTor] = rtde.movej(target,'joint')
Want joint position and joint acceleration and input 'joint' positions	[~,jointPos,~, jointAcc] = rtde.movej(target,'joint')
If you want to set acceleration, velocity or blend radius input variables. a = tool acceleration v = tool speed r = blend radius You will still need to provide a value t = 0. Remember if t > 0 it will override a and v	[pose] = rtde.movej(target,'joint',a,v,t,r) **If via_point and target positions are a pose, you need to use:** [pose] = rtde.movej(target,'pose',a,v,t,r)
If you want to set a value for time: e.g. t = 5 Again you will still need to input a and v, but they can be set to 0.	[pose] = rtde.movej(target,'joint',a,v,t,r) **If via_point and target positions are a pose, you need to use:** [pose] = rtde.movej(target,'pose',a,v,t,r)

movel : Move to position (linear in tool space)

[poses,joints,jointVelocities,jointAccelerations,torques] = movel(target,jointOrPose,a,v,t,r)

PARAMETERS:

- **Target =** Joint position or pose.
- **jointOrPose** = "joint" or "pose" to provide joint inputs or pose inputs
- **a** = joint acceleration (m/s^2) (default = 1.4)
- **v** = joint speed (m/s) (default = 1.05)
- **t =** time (s) (default = 0). If specified it will ignore the **a** and **v** values
- **r** = blend radius (of target pose) (m) (default = 0)

movel : example usage

target = [-588.53, -300, 200, 2.2214, -2.2214, 0.00];

	command
Output pose and input xyzrpy	pose = rtde.movel(target)
Output pose and joint position and input xyzrpy	[pose,jointPos] = rtde.movel(target)
Output pose, joint position, joint velocity	[pose,jointPos,jointVel] = rtde.movel(target)
Output pose, joint position, joint velocity, joint acceleration	[pose,jointPos,jointVel, jointAcc] = rtde.movel(target)
Output pose, joint position, joint velocity, joint acceleration, joint torque	[pose,jointPos,jointVel, jointAcc, jointTor] = rtde.movel(target)
Only want joint position	[~,jointPos] = rtde.movel(target)
Only want joint velocity	[~,jointPos,~] = rtde.movel(target)
Want joint position and joint acceleration	[~,jointPos,~, jointAcc] = rtde.movel(target)

movel : example usage

target = [-2.4670 -2.1287 -1.6192 -0.9645 1.5708 -2.4670];

	command
Output pose and input joint position	pose = rtde.movel(target,'joint')
Output pose, joint position, joint velocity, joint acceleration, joint torque and input joint positions	[pose,jointPos,jointVel, jointAcc, jointTor] = rtde.movel(target,'joint')
Want joint position and joint acceleration and input 'joint' positions	[~,jointPos,~, jointAcc] = rtde.movel(target,'joint')
If you want to set acceleration, velocity or blend radius input variables. a = tool acceleration v = tool speed r = blend radius You will still need to provide a value t = 0. Remember if t > 0 it will override a and v	[pose] = rtde.movel(target,'joint',a,v,t,r) **If via_point and target positions are a pose, you need to use:** [pose] = rtde.movel(target,'pose',a,v,t,r)
If you want to set a value for time: e.g. t = 5 Again you will still need to input a and v, but they can be set to 0.	[pose] = rtde.movel(target,'joint',a,v,t,r) **If via_point and target positions are a pose, you need to use:** [pose] = rtde.movel(target,'pose',a,v,t,r)

movep : Blend circular and move linear (in tool space)

Definition: [poses,joints,jointVelocities,jointAccelerations,torques] = movep(obj,target,jointOrPose,a,v,r)

Example use case is exactly the same as movej and movel. Please refer to those.

Due to the safety limits on the UR5e, it might not work as it tends to try to go at a velocity about the limit. But try it out and see what happens!

movec : Circular Move: Move to position (circular in tool-space)

TCP moves on the circular arc segment from current pose, through pose_via to pose_to. Accelerates to and moves with constant tool speed v. Use the mode parameter to define the orientation interpolation.

Definition:

[poses,joints,jointVelocities,jointAccelerations,torques] = movec(via_point,pose_to,jointOrPose,a,v,r,mode)

movec : Circular Move: Move to position (circular in tool-space)

[poses,joints,jointVelocities,jointAccelerations,torques] = movec(via_point,pose_to,jointOrPose,a,v,r,mode)

PARAMETERS:

- **pose_via** = path point Pose_via, can also be specified as joint positions, then forward kinematics is used to calculate the corresponding pose.
- **pose_to** = arget pose Pose_to can also be specified as joint positions, then forward kinematics is used to calculate the corresponding pose.
- **jointOrPose**= "joint" or "pose" to provide joint inputs or pose inputs
- **a** = tool acceleration (m/s^2) (default = 1.2)
- **v** = tool speed (m/s) (default = 0.25)
- **r** = blend radius (of target pose) (m) (default = 0.05)
- **mode = 0**: Unconstrained mode. Interpolate orientation from current pose to target pose (pose_to)
- **mode = 1**: Fixed mode. Keep orientation constant relative to the tangent of the circular arc(starting from current pose)
- (mode default = 1)

movec : example usage

via_point = [-588.53, -300, 400, 2.2214, -2.2214, 0.00];

target = [-588.53, -300, 200, 2.2214, -2.2214, 0.00];

	command
Output pose and input xyzrpy	pose = rtde.movec(via_point,target)
Output pose and joint position and input xyzrpy	[pose,jointPos] = rtde.movec(via_point,target)
Output pose, joint position, joint velocity	[pose,jointPos,jointVel] = rtde.movec(via_point,target)
Output pose, joint position, joint velocity, joint acceleration	[pose,jointPos,jointVel, jointAcc] = rtde.movec(via_point,target)
Output pose, joint position, joint velocity, joint acceleration, joint torque	[pose,jointPos,jointVel, jointAcc, jointTor] = rtde.movec(via_point,target)
Only want joint position	[~,jointPos] = rtde.movec(via_point,target)
Only want joint velocity	[~,jointPos,~] = rtde.movec(via_point,target)
Want joint position and joint acceleration	[~,jointPos,~, jointAcc] = rtde.movec(via_point,target)

movec : example usage

via_point = [-588.53, -300, 400, 2.2214, -2.2214, 0.00];

target = [-588.53, -300, 200, 2.2214, -2.2214, 0.00];

x,y,z are in mm and **r,p,y** are in radians

	command
Output pose and input joint position	pose = rtde.movec(via_point,target)
Output pose and joint position and input xyzrpy	[pose,jointPos] = rtde.movec(via_point,target)
Output pose, joint position, joint velocity	[pose,jointPos,jointVel] = rtde.movec(via_point,target)
Output pose, joint position, joint velocity, joint acceleration	[pose,jointPos,jointVel, jointAcc] = rtde.movec(via_point,target)
Output pose, joint position, joint velocity, joint acceleration, joint torque	[pose,jointPos,jointVel, jointAcc, jointTor] = rtde.movec(via_point,target)
Only want joint position	[~,jointPos] = rtde.movec(via_point,target)
Only want joint velocity	[~,jointPos,~] = rtde.movec(via_point,target)
Want joint position and joint acceleration	[~,jointPos,~, jointAcc] = rtde.movec(via_point,target)

movec : example usage

```
via_point = [-2.4670   -1.9549   -1.3324   -1.4250    1.5708   -2.4670];
target = [-2.4670   -2.1287   -1.6192   -0.9645    1.5708   -2.4670];
```

	command
Output pose and input joint position	pose = rtde.movec(via_point,target,'joint')
Output pose, joint position, joint velocity, joint acceleration, joint torque and input joint positions	[pose,jointPos,jointVel, jointAcc, jointTor] = rtde.movec(via_point,target,'joint')
Want joint position and joint acceleration and input 'joint' positions	[~,jointPos,~, jointAcc] = rtde.movec(via_point,target,'joint')
If you want to set acceleration, velocity or blend radius input variables. a = tool acceleration v = tool speed r = blend radius	[pose] = rtde.movec(via_point,target,'joint',a,v,r) **If via_point and target positions are a pose, you need to use:** [pose] = rtde.movec(via_point,target,'pose',a,v,r)

Other functions

Get actual tool pose (x,y,z,r,p,y) (m and radians)	pose = rtde.actualPosePositions()
Get target tool pose	pose = rtde.targetPosePositions()
Get actual joint position	jointPos = rtde.actualJointPositions()
Get target joint positions	jointPos = rtde.targetJointPositions()
Plot tool path (poses)	rtde.drawPath(listOfPoses)
Plot joint positions	rtde.drawJointPositions(listOfJointPositions)
Plot joint velocities	rtde.drawJointVelocities(listOfJointVelocities)
Plot joint accelerations	rtde.drawJointAccelerations(listOfJointAccelerations)
Plot joint torques	rtde. drawJointTorques(listOfTorques)

Robot Mode

mode = rtde.checkRobotMode()

% RETURN VALUES:
%-1 : ROBOT_MODE_NO_CONTROLLE
% 0 : ROBOT_MODE_DISCONNECTED
% 1 : ROBOT_MODE_CONFIRM_SAFETY
% 2 : ROBOT_MODE_BOOTING
% 3 : ROBOT_MODE_POWER_OF
% 4 : ROBOT_MODE_POWER_ON
% 5 : ROBOT_MODE_IDLE
% 6 : ROBOT_MODE_BACKDRIVE
% 7 : ROBOT_MODE_RUNNING
% 8 : ROBOT_MODE_UPDATING_FIRMWARE

Safety mode

safety = rtde.checkSafetyMode()

% RETURN VALUES:
% 1 : SAFETY_MODE_NORMAL
% 2 : SAFETY_MODE_REDUCED
% 3 : SAFETY_MODE_PROTECTIVE_STOP
% 4 : SAFETY_MODE_RECOVERY
% 5 : SAFETY_MODE_SAFEGUARD_STOP
% 6 : SAFETY_MODE_SYSTEM_EMERGENCY_STOP
% 7 : SAFETY_MODE_ROBOT_EMERGENCY_STOP
% 8 : SAFETY_MODE_VIOLATION
% 9 : SAFETY_MODE_FAULT
% 10 : SAFETY_MODE_VALIDATE_JOINT_ID
% 11 : SAFETY_MODE_UNDEFINED_SAFETY_MODE

Debugging

- If you get an error saying “Unrecognised function or variable ‘rtde’”, that means you have not correctly set the path to the location of the rtde folder in matlab.
  - Please follow instructions here: https://github.com/rag-h/mtrn4230_course_development

```
>> example0
Unrecognized function or variable 'rtde'.

Error in example0 (line 10)
rtde = rtde(host,port);
```