

# Mobile Tutorial 1

## *Mobile Tutorial 1 project build order*

- A) If you have built all the other projects from the previous solutions step by step, you just need to build the projects in Column A.
- B) If you haven't built any of the previous tutorial solutions, you need to build all included projects in the order shown in Column B.

### Column A

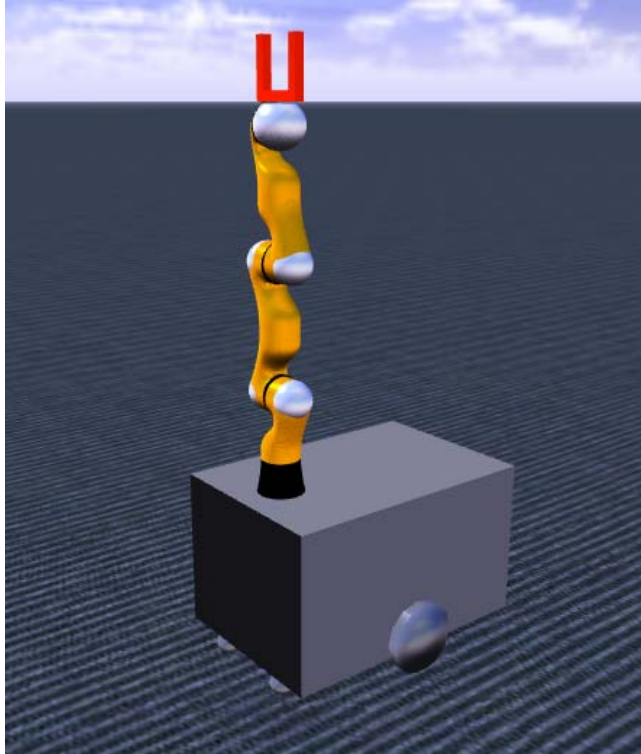
1. SimulatedDifferentialDrive
2. KUKASimulatedLBR3Gripper
3. KUKAMobileTutorial1Simulation
4. KUKATutorial1Dashboard

### Column B

1. Util
2. KUKACommandTypes
3. ArticulatedArm (in abstract services)
4. DriveDifferentialTwoWheel (in abstract services)
5. KUKARobotTool (in abstract services)
6. KUKAUniversalMotionPlanning (in abstract services)
7. Transformation
8. SimulatedLBR3Arm
9. KUKATutorial3MotionPlanning
10. SimulatedDifferentialDrive
11. KUKASimulatedLBR3Gripper
12. KUKAMobileTutorial1Simulation
13. KUKATutorial1Dashboard

## ***Adding platform and gripper***

In this tutorial, we add two actors to the scene: a mobile platform and a Gripper.



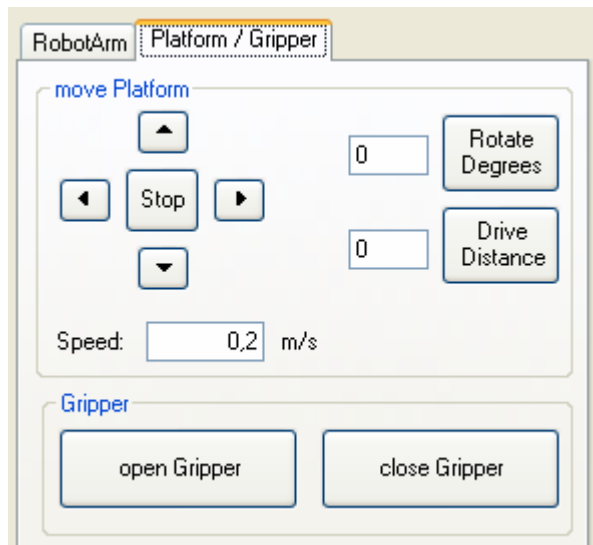
For controlling those entities you will find a new tab on the dashboard: "Platform/Gripper".

*Platform:*

Use the buttons with the arrows to command the platform to either move continuously forward or backward, or to rotate until a different command has been issued. The speed for these movements can be set in the edit box below the cursor buttons.

If the platform has to move a certain distance you can set this distance in the editbox. If the platform has to rotate about a specific angle, you can set this angle in the editbox next to the 'Rotate Degrees' button.

NOTE: Thanks to the physics engine, you will see that the degrees or length that you have specified will not be hit exactly. Actually sometimes the result can differ quite substantially from your expectation. This has to do with the slippage that occurs during the acceleration of the platform. This should be considered a very positive side of this simulation environment, since it approximates the problems you will run into in the real world.



*Gripper:*

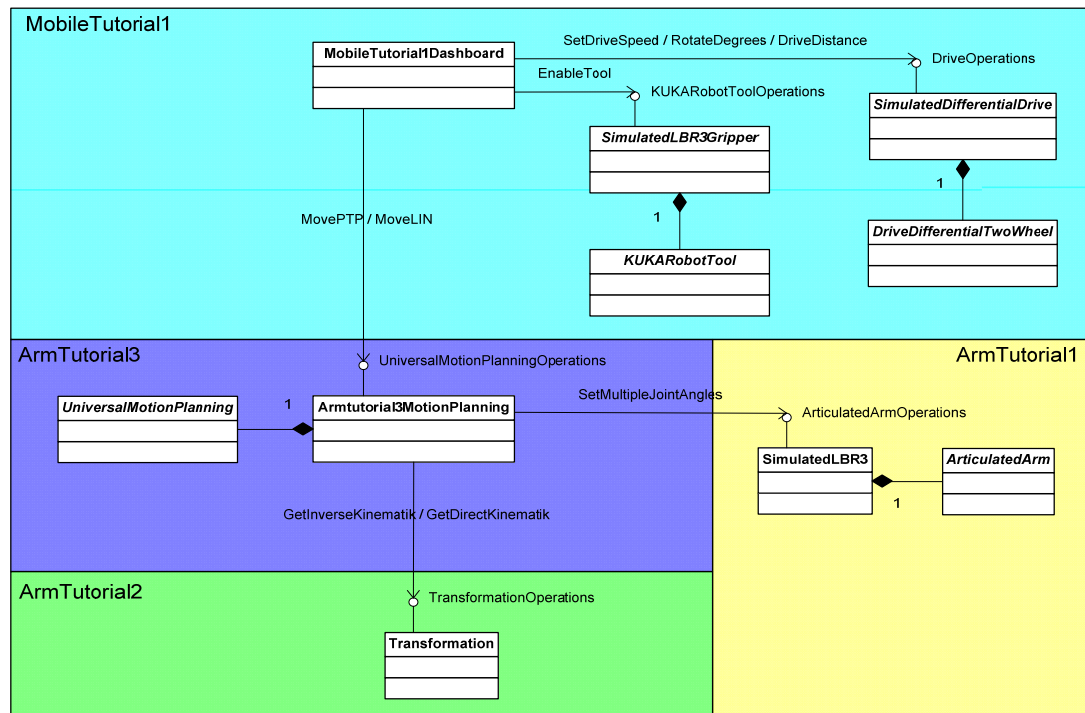
Use the 'open Gripper' button or the 'close Gripper' button to send a command to the gripper entity.

The Gripper itself is created by using KUKA arm link entities, where the joint property has been set to model a linear axis instead of a rotational axis.

## Coding for mobile Tutorial 1:

Since we do not orchestrate the issued commands through a specialized service in this tutorial, the commands are sent directly from the dashboard to the corresponding services controlling simulation entities.

Mobile Tutorial 1 Services Overview:



New services for mobile tutorial 1:

- MobileTutorial1Dashboard: User Interface with an extra page for controlling platform and gripper
- SimulatedDifferentialDrive: Controls the platform entity
- DriveDifferentialTwoWheel: Constitutes the Differential Drive Interface for the platform
- SimulatedLBR3Gripper: Controls the gripper entity
- KUKARobotTool: Constitutes the KUKA Tool Interface for the gripper

The diagram also shows you the three main command connections coming from the Dashboard:

**MovePTP/MoveLIN:** Commands to the robot: Sends axis values or cartesian coordinates directly to the motion planning service, which is responsible to move the simulated LBR.

EnableTool/DisableTool: Commands to the tool: The 'EnableTool' command closes the Gripper. The 'DisableTool' command opens the Gripper.

SetDriveSpeed/...: Commands to the platform: These commands control the platform entity

## Issuing commands to the platform:

If, as an example, a forward motion command has been selected, it is first processed in the GUI handler for the corresponding GUI button. A new instance of a "OnMovePlatform" command is sent to the eventport of the dashboard service.

```
private void btn_Forward_Click(object sender, EventArgs e)
{
    try
    {
        _eventsPort.Post(new OnMovePlatform(this, MoveDirections.Forward, float.Parse(txt_Speed.Text), 0));
    }
    catch (FormatException ex)
    {
        MessageBox.Show(ex.Message, "An error occured", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

In the event handler for this specific GUI event, the 'SetDrivePowerRequest' method of the \_drivePort is executed.

```
/// <summary>
/// moves the platform
/// </summary>
/// <param name="move">movement parameters</param>
/// <returns>tasks to perform</returns>
IEnumerator<ITask> OnMovePlatformHandler(OnMovePlatform move)
{
    .
    .
    diffdrive.SetDriveSpeedRequest powerReq = new diffdrive.SetDriveSpeedRequest(leftWheel, rightWheel);
    yield return Arbiter.Choice(
        _drivePort.SetDrivePower(powerReq),
        delegate(DefaultUpdateResponseType resp) { },
        delegate(Fault f) { }
    );
    .
    .
}
```

The \_drivePort is the variable for the DriveOperations port of the partner 'diffdrive'.

```
/// <summary>
/// Partner: Differential Drive service
/// </summary>
[Partner("diffdrive", Contract=diffdrive.Contract.Identifier, CreationPolicy=PartnerCreationPolicy.UseExisting)]
private diffdrive.DriveOperations _drivePort = new diffdrive.DriveOperations();
```

Sending the command to this 'diffdrive' partner causes the simulated platform to move accordingly.

## Issuing commands to the gripper:

By pressing the 'open gripper' button or the 'close gripper' button, a new 'OnChangeGripper' message is generated. This message is processed in the OnChangeGripperHandler, which calls either 'EnableTool' or 'DisableTool' from the GripperOperationsPort.

```
/// <summary>
/// changes the state of the gripper
/// </summary>
/// <param name="grip">gripper parameter</param>
/// <returns>tasks to perform</returns>
IEnumerator<ITask> OnChangeGripperHandler(OnChangeGripper grip)
{
    //close the gripper
    if (_gripperIsOpen)
    {
        yield return Arbiter.Choice(
            _gripperOperations.EnableTool(new gripper.EnableToolRequest()),
            delegate(DefaultUpdateResponseType resp) { },
            DefaultFaultHandler
        );
        WinFormsServicePort.FormInvoke(delegate()
        {
            _gui.changeGripperButtonText(true);
        });
    }
    //open the gripper
    else
    {
        .
        .
        .
    }
}
```

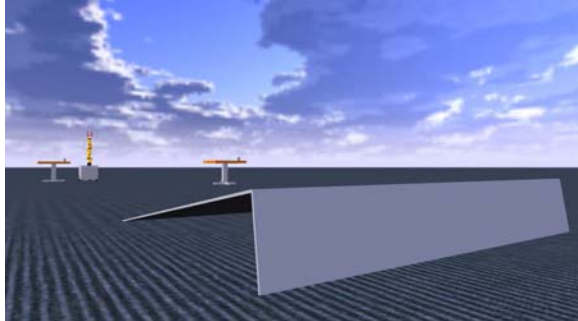
The GripperOperationsPort is defined as a partner of the dashboard service.

```
/// <summary>
/// Partner: Gripper Service
/// </summary>
[Partner("Gripper", Contract = gripper.Contract.Identifier, CreationPolicy = PartnerCreationPolicy.UseExisting)]
private gripper.KukarobottoolOperations _gripperOperations = new gripper.KukarobottoolOperations();
```

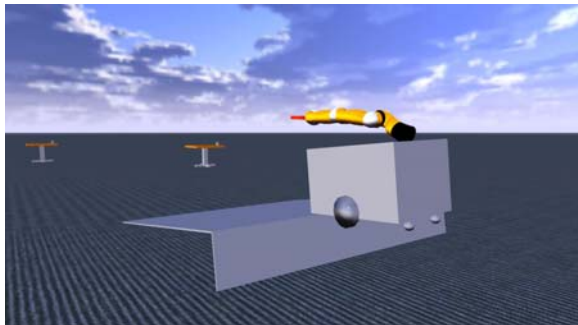
Sending the command to this 'Gripper' partner causes the simulated gripper to react accordingly.

## ***Fun Corner***

Now that the LBR arm is on wheels, we couldn't get around building a ramp for it...



With four meters per second, you can get already good air time...



...and fortunately the simulated hardware is not expensive at all compared to the real hardware...

