

**INTRODUCTION TO ROBOTICS**

**(MTS -417)**

**DE-44 Mechatronics Syndicate– C**

**Lab Report 3**

**Name of members:**

**NC** M. Ahmed Hussain Babar (**427781)**

**PC** Abdul Haseeb Zahid (**432219)**

**PC** Rehan Shahid (**432283)**

**Submitted to**: LE Hamza Sohail

**Task 1: Velocity Control**

Weightage: 3% out of 100 of Lab

TASK  
 Your task for this Lab is to implement Lab 5 (conveyor belt system) from "Lab 5" already uploaded on LMS.   
Few changed that all groups have to implement.

(i) Take group leader registration number "123456". Add third, fourth and fifth digit. These represent your number of cuboids that will be on top of conveyor moving by conveyor belt.

(ii) It is upto you to adjust either cuboids or conveyor as per convenience.

(iii) When sensor detects a cuboid, conveyor belt should stop for 2 seconds only for each cuboid and after that cuboid should fall on the ground.

(iv) You can use sim.getSimulationTime( ) or you can use sim.wait( ). But sim.wait ( ) only works in threaded script.

|  |
| --- |
| **Lua Script:** |
| function sysCall\_init()  sim = require('sim')  prismaticJoint = sim.getObjectHandle('Prismatic\_joint')  targetVelocity = 0.1  -- do some initialization here  end  function sysCall\_actuation()  sim.setJointTargetVelocity(prismaticJoint, targetVelocity)  end  function sysCall\_sensing()  -- put your sensing code here  end  function sysCall\_cleanup()  -- do some clean-up here  end  -- See the user manual or the available code snippets for additional callback functions and details |

**Simulation Snippet:**

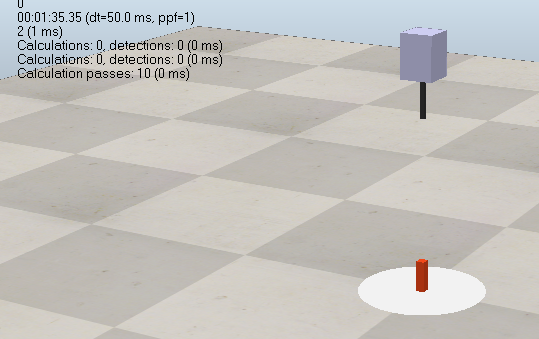


Figure 1

**Task 2: Position Control**

|  |
| --- |
| **Lua Script:** |
| function sysCall\_init()  sim = require('sim')  prismaticJoint = sim.getObjectHandle('Prismatic\_joint')  targetPosition = 0.2  -- do some initialization here  end  function sysCall\_actuation()  sim.setJointTargetPosition(prismaticJoint, targetPosition)  end  function sysCall\_sensing()  -- put your sensing code here  end  function sysCall\_cleanup()  -- do some clean-up here  end  -- See the user manual or the available code snippets for additional callback functions and details |

**Simulation Snippet:**

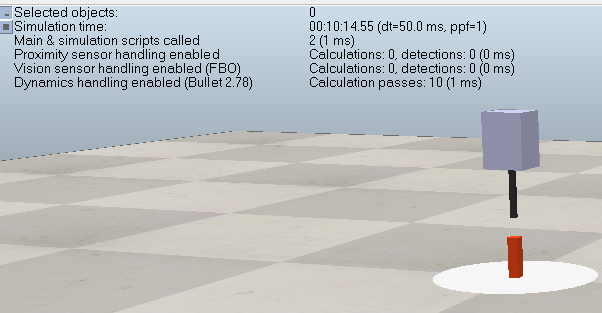


Figure 2

**Results:**

* **The prismatic joint moved linearly along its defined axis under both velocity and position control.**
* **In velocity control mode, the cuboid moved at a steady constant speed according to the target velocity value.**
* **In position control mode, the cuboid smoothly reached the specified target position and then stopped precisely.**
* **Changing the target values directly affected motion behavior — higher velocity made the cuboid move faster, while higher position values increased the displacement.**
* **The control response was stable and linear, with no oscillations or abrupt motion.**

**Discussion**:

* The experiment confirmed that the same control functions used for revolute joints can be applied to prismatic joints, with the difference being **linear translation** instead of **rotation**.
* **Velocity control** is suitable for continuous linear movement, while **position control** provides accurate displacement control for specific tasks.
* This behavior is similar to real-world **linear actuators** used in robots for precise pushing or sliding motions.
* The simulation demonstrated how **joint control parameters** (velocity, position) directly influence motion performance and system stability.

**Conclusion**:

The lab successfully demonstrated velocity and position control of a prismatic joint connected to a cuboid. Velocity control achieved consistent linear motion, while position control accurately reached target positions. The setup and script can be extended for more complex robotic tasks.