[Company name]  [Company address]

Library Bot documentation

Contents

[To do 2](#_Toc101351073)

[Code structure 3](#_Toc101351074)

# To do

* Get DH parameters for robot and make class in Matlab (Nhan)
* Setting up environment (Nick)
* Setting up functions and classes (Patrick)
* Once we have our models on Solidworks, create a class for a rectangular prism obstacle. This will be used for anything the robot can collide with (table it is on, and shelves). The class will plot the obstacle (which is static), and also create a triangle mesh of the rectangular prism (see week 5 lab question 2 and 3 solution videos).

# Code structure

Hans cute class:

|  |  |  |
| --- | --- | --- |
| Create robot (constructor with base location) | **Done** | HansCute() |
| Set base location (alternative to above) | **------** | Already a model.base() function by default |
| Get current end effector position (fkine) | ------ | model.fkine(q) |
| Grab function (for end effector) | ------ | Self.gripperBool = true/false |
| Get grab status | ------ | Self.griperBool (returns true or false) |
| Get position of EVERY joint (useful for collision detection) |  |  |
| Set joint limits | ------ | Done within class constructor, in DH params |
| Collision detects with surface (input surface [by vertexes?]) (iterative to check every line of each robot arm, with each surface) |  |  |
| Stop robot movement |  | Function which stops robot movement. Used for both collision and estop. Therefore, one variable is for the stop function, and another indicates whether it was a collision or estop. |
| Move from point A to B in a linear fashion. MUST obey some predefined joint limits (defined within the function), and constantly be checking for collisions. Use RMRC |  |  |

Graphical user interface, text, application, letter

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated