```
"""
Code from example-code
used as communication with robot
Author: Erik Billing (billing@cs.umu.se)
Updated by Ola Ringdahl 204-09-11
"""

MRDS_URL = 'localhost:50000'

import httplib, json, time
from math import sin, cos, pi, atan2

HEADERS = {"Content-type": "application/json", "Accept": "text/json"}

mrds = httplib.HTTPConnection(MRDS_URL)


class UnexpectedResponse(Exception): pass


def postSpeed(angularSpeed, linearSpeed):
    """Sends a speed command to the MRDS server"""
    params = json.dumps({'TargetAngularSpeed': angularSpeed, 'TargetLinearSpeed': linearSpe
ed})
    mrds.request('POST', '/lokarria/differentialdrive', params, HEADERS)
    response = mrds.getresponse()
    data=response.read()
    status = response.status
    # response.close()
    if status == 204:
        return data
    else:
        raise UnexpectedResponse(response)

def getLaser():
    """Requests the current laser scan from the MRDS server and parses it into a dict"""
    mrds.request('GET', '/lokarria/laser/echoes')
    response = mrds.getresponse()
    if (response.status == 200):
        laserData = response.read()
        return json.loads(laserData)
    else:
        return response

def getPose():
    """Reads the current position and orientation from the MRDS"""
    mrds.request('GET', '/lokarria/localization')
    response = mrds.getresponse()
    if (response.status == 200):
        poseData = response.read()
        return json.loads(poseData)
    else:
        return UnexpectedResponse(response)


def bearing(q):
    return rotate(q, {'X': 1.0, 'Y': 0.0, "Z": 0.0})


def rotate(q, v):
    return vector(qmult(qmult(q, quaternion(v)), conjugate(q)))


def quaternion(v):
    q = v.copy()
    q['W'] = 0.0;
    return q
```

```
def vector(q):
    v = {}
    v["X"] = q["X"]
    v["Y"] = q["Y"]
    v["Z"] = q["Z"]
    return v


def conjugate(q):
    qc = q.copy()
    qc["X"] = -q["X"]
    qc["Y"] = -q["Y"]
    qc["Z"] = -q["Z"]
    return qc


def qmult(q1, q2):
    q = {}
    q["W"] = q1["W"] * q2["W"] - q1["X"] * q2["X"] - q1["Y"] * q2["Y"] - q1["Z"] * q2["Z"]
    q["X"] = q1["W"] * q2["X"] + q1["X"] * q2["W"] + q1["Y"] * q2["Z"] - q1["Z"] * q2["Y"]
    q["Y"] = q1["W"] * q2["Y"] - q1["X"] * q2["Z"] + q1["Y"] * q2["W"] + q1["Z"] * q2["X"]
    q["Z"] = q1["W"] * q2["Z"] + q1["X"] * q2["Y"] - q1["Y"] * q2["X"] + q1["Z"] * q2["W"]
    return q


def getBearing():
    """Returns the XY Orientation as a bearing unit vector"""
    return bearing(getPose()['Pose']['Orientation'])
```