## **Robot Environment** - Spyder robot

*Simulator:* Pybullet
*Reinforcement environment:* OpenAi Gym

Goal: Make the Arakno robot to learn how to walk and reach an endpoint in a 2D plane.
Env: arakno bot, flat plane, endpoint position

**AraknoEnv:**

This class represents the control of the robot. It's structured in a way that it makes compatible with the Gym environment -> methods:

- __init__ -> initialize the env: adding robot to flat ground
- reset -> reset the simulation, respawn the robot at the start point
- step -> return observation, reward, done, info
- close -> disconnect Pybullet

*Action and Observation spaces:*

- *Action space:* will consist of a number in the range of $[-1,1]$ which will apply the torque to the specified joint.
- *Observation space:* continuous space, range [-inf,+inf]. Each leg has num_joints + 1 observation dim, representing the joint angle for each joint + 1 for the foot position. Include 6 dim to represent the position and orientation in 3D space. ( *shape=(self.num_legs * (self.num_joints + 1) + 6,))*

*Robot control description:*

- *observations:* the observations of the **environment are the generalized coordinates and generalized velocities** of each joint. Its a vector of dim 37, where 19 dim for the angle value of each joint and 18 for the velocity-angle of each joint. 19 because the orientation of the body base it's expressed in quaternion
- *is_alive:* check if the robot is fallen down. For doing that calculate where is the position of the COM of the body base, check if beyond a threshold height
- *check_done*: check if the experiment is done by checking the following conditions:
    - reached the endpoint

    - fallen on the ground
    - maximum number timesteps reached

- *compute_progress:* compute the distance between the current position and the endpoint target. The formula used is the following ones:

    - $P := \sqrt{x_{\Delta_t}^2 + y_{\Delta_t}^2} \quad P_{t-1} := \sqrt{x_{\Delta_{t-1}}^2 + y_{\Delta_{t-1}}^2} \quad P_t - P_{t-1}$
    - This encourage the robot to move forward and make progress
- *compute_reward:* the reward as a weighted sum of stability, progress, and still alive. Where the stability is calculated with respect to the COM of the body base, and it's penalized if the robot falls on the ground.

Additional:

- camera joint: in the case its need to have visual data for learn to walk(for now no, maybe on an uneven terrain later)
- function *render():* class-method, enables three different type of view attached to the camera of the robot: grayscale, RGB, and depth