document

Actat

1　ロボット製作の動機

このロボットの製作に取り組むのは，歩行ロボット製作を経験するためである．最終的には二足歩行ロボットの製作を目指しているが，6 自由度の脚の設計は難易度が高いと判断し，今回は 3 自由度の脚を 4 つ備えた四足歩行ロボットを製作することにした．

このロボットの製作では以下の内容を経験することを期待している．

- 自宅でのロボット製作
- サーボモータを用いた機構の設計
- 最低限の電気回路の取扱い
- ROS を用いたプログラミング
- 歩行ロボットの制御

2　機械設計・製作

設計・製作したロボットを図 1 に示す．脚は前後左右に鏡像になっている同一の機構とした．サーボモータは研究室で使用しているモータと合わせて近藤科学株式会社の B3M-SC-1170-A を用いた．CIT Brains が公開している B3M モータを用いた二足歩行ロボットを参考に，関節間の距離は 100 mm にした．

胴体中央下部に株式会社アールティの USB 出力 9 軸 IMU センサモジュールを搭載した．センサの位置をロボットのベースリンクと一致させて計算を簡単にする意図がある．胴体上部前方に取り付けられた板にサーボモータと PC の間の通信のための基板を固定した．

3　電装

ロボット全体で 12 個のモータが使われている．モータとパソコンは RS485USB/シリアル変換アダプターを介して通信する．直列に接続できるモータの数は限られているので XH コネクター用ハブ typeA によって各脚ごとの系統に分けて接続した．配線の様子を図 2 に示す．

XH コネクター用ハブには 5 つのコネクタがあり，すべて並列に接続される．このハブはロ

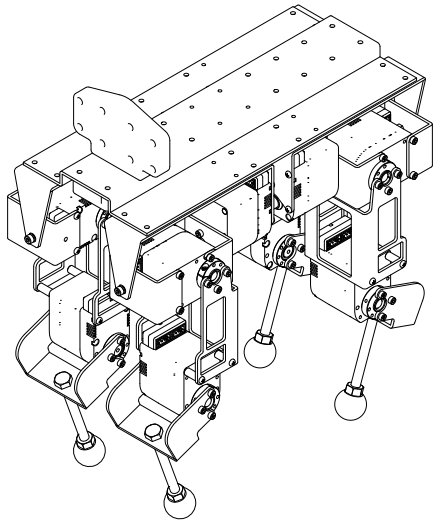

図 1　設計・製作したロボット

ボット前方に横向きに取り付けられており，中央のコネクタを RS485USB/シリアル変換アダプターに接続し，左右のコネクタをそれぞれ脚のモータと接続した．脚の 3 つのモータは Hip flexion/extension, Hip ab/adduction, knee の順に接続された．Hip のモータは機械的な接続とは逆の順序である．

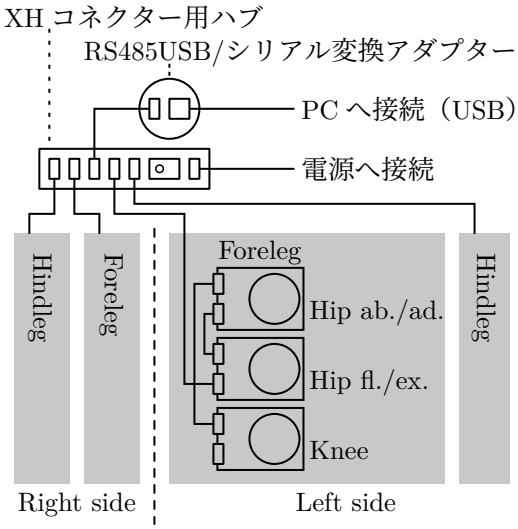

図 2　配線の様子

サーボモータは配線に先立って ID の書き込みを行った．書き込んだ ID は表 1 に示すとおりである．

表 1　ID と関節の対応

id	脚	関節
0	左前	Hip ab/adduction
1	左前	Hip flexion/extension
2	左前	knee
3	右前	Hip ab/adduction
4	右前	Hip flexion/extension
5	右前	knee
6	左後	Hip ab/adduction
7	左後	Hip flexion/extension
8	左後	knee
9	右後	Hip ab/adduction
10	右後	Hip flexion/extension
11	右後	knee

4　Package の作成

別のディレクトリで`$ ros2 pkg create --build-type ament_cmake --node-name quadruped_takahashi quadruped_takahashi` を実行して package.xml, CMakeLists.txt, quadruped_takahashi.cpp を作成した．その後，作成したファイルをこのリポジトリ内へ移動した．

5　モータとの通信

モータとの通信には Actat/kondo_b3m_ros2 を用いる．motor_class ブランチ[1)]に合わせて launch ファイルを用意し，通信ができることを確認する．以下に launch ファイルのうち，kondo_b3m_ros2 のノードの設定に関する部分を抜粋する．軸の方向を反転しているのは，後述する座標系に合わせるためである．

```
kondo_b3m_ros2_node = Node(
  package='kondo_b3m_ros2',
  executable='kondo_b3m',
  remappings=[('b3m_joint_state', 'joint_states')],
  parameters=[{'motor_list': [
    "{'id': 0, 'name': 'lf0', 'direction': False}",
    "{'id': 1, 'name': 'lf1', 'direction': False}",
```

1)　このプロジェクトでうまく動作することを確認したら master にする予定である．

```
        "{'id': 2, 'name': 'lf2', 'direction': False}",
        "{'id': 3, 'name': 'rf0', 'direction': False}",
        "{'id': 4, 'name': 'rf1'}",
        "{'id': 5, 'name': 'rf2'}",
        "{'id': 6, 'name': 'lh0'}",
        "{'id': 7, 'name': 'lh1', 'direction': False}",
        "{'id': 8, 'name': 'lh2', 'direction': False}",
        "{'id': 9, 'name': 'rh0'}",
        "{'id': 10, 'name': 'rh1'}",
        "{'id': 11, 'name': 'rh2'}"
      ]}],
    )
```

あるウインドウで`$ ros2 launch quadruped_takahashi launch.py` を実行して起動し，別のウインドウで`$ ros2 topic echo /joint_states` することで各関節の角度・角速度が出力されていることを確認した．

6 URDF ファイルの作成

URDF ファイルの作成にあたってロボットの関節に関する座標系を設定する．初めに左前脚を考える（図 3）．2 つの Hip の関節は軸が交わるように作られており，この交点を脚の付け根とする．脚の付け根を原点としてロボットの前方に x 軸，左方向に y 軸，鉛直上方向に z 軸をとった座標系を Σ_{lf0} とする．Σ_{lf0} の x 軸は Hip ab/adduction の関節軸と一致しており，この軸周りに θ_{lf0} 回転した座標系を Σ_{lf1} とする．θ_{lf0} は x 軸の方向を正とする．Σ_{lf1} の y 軸は Hip flextion/extension の関節軸と一致しており，この軸周りに Σ_{lf1} 回転した座標系を Σ_{lf2} とする．Σ_{lf1} は y 軸の方向を正とする．この座標系 Σ_{lf2} は腿のリンクと対応しており，腿は $-z$ の方向にのびている．膝関節は Σ_{lf2} において $[0, 0, -0.1]\,\mathrm{m}$ の位置に存在する．この位置を原点とし，y 軸周りに θ_{lf2} 回転した座標系を Σ_{lf3} とする．この座標系は脛のリンクと対応しており，$[0, 0, -0.1]\,\mathrm{m}$ の位置が足の球の中心になる．

他の脚にも同様に 3 つの関節角と 4 つの座標系を定義する．すべて座標軸は直立したときに，x 軸が前方，y 軸は左方を向くように設定する．関節角の向きは x, y 軸の正方向によって定める．

ロボットの中心となる base_link の位置は，4 つの hip joint の中心に定める．座標系の向きは x 軸が前方，y 軸は左方，z 軸が鉛直上方である．

URDF ファイルを作成して，CMakeLists.txt に追記した．rviz の config ファイルも同様に CMakeLists.txt に追記した．launch.py を編集し，rviz を起動して/joint_states の関節角をモデルの表示に反映させる．base_link を固定して表示させることができた（図 4）．

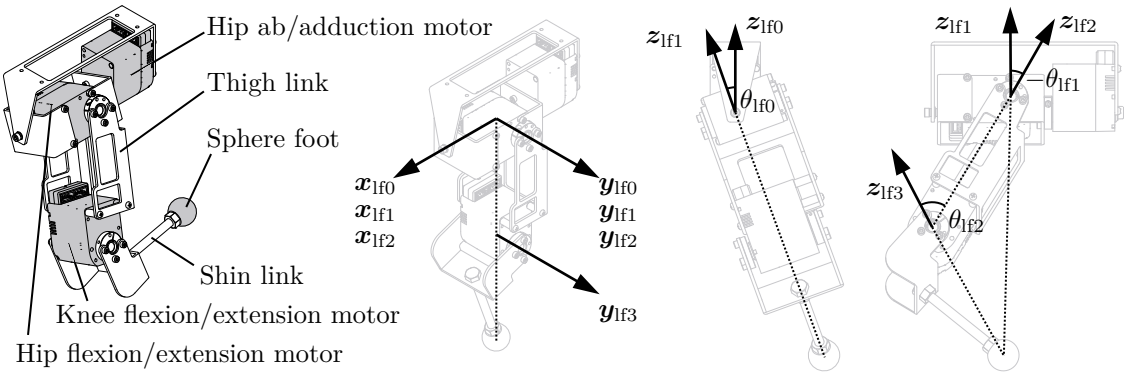

図 3　左前脚の座標系の設定．脚は 3 つの関節軸を持っておりそれぞれが独立に回転する．脚を垂直に伸ばした状態で 2 つの尻関節の交点と足の球面の中心を結ぶ線分上に各座標系の原点を設定する．それぞれの関節の回転角を $\theta_{\mathrm{lf0}}, \theta_{\mathrm{lf1}}, \theta_{\mathrm{lf2}}$ とする．

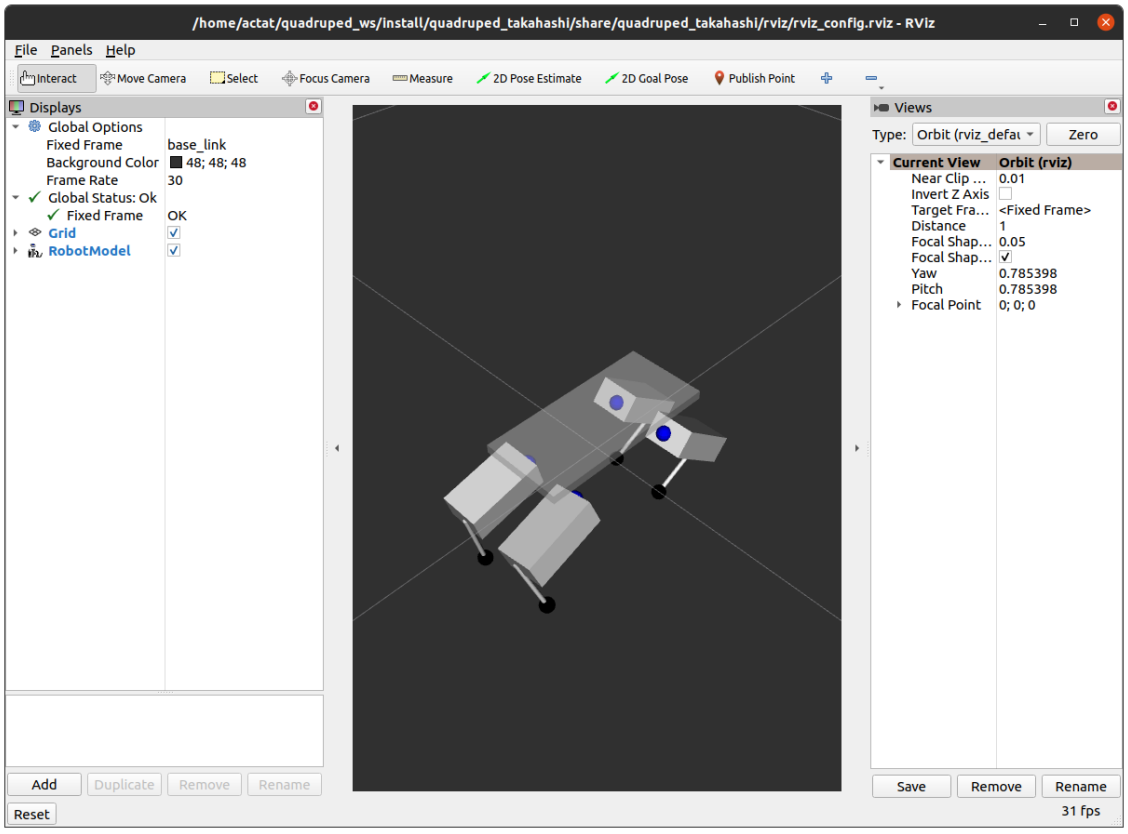

図 4　URDF ファイルを読み込んで表示する rviz の画面