# Architecture

The architecture is denoted below. On the mechanical side, we have two actuators driven by a servo (mainly due to space restrictions) and four actuators driven by a stepper/rotary encoder combination.

The servos (Herkulex DRS-101) are controlled directly by the controller board with an ATMega644 on board via a serial interface. The steppers do not have an internal feedback loop, so we need rotary encoders detecting the absolute angle of the joint and allowing to implement feedback controllers. Depending on the actuator, the steppers provide a torque between 26Ncm (elbow) and 3,1Nm (upperarm).

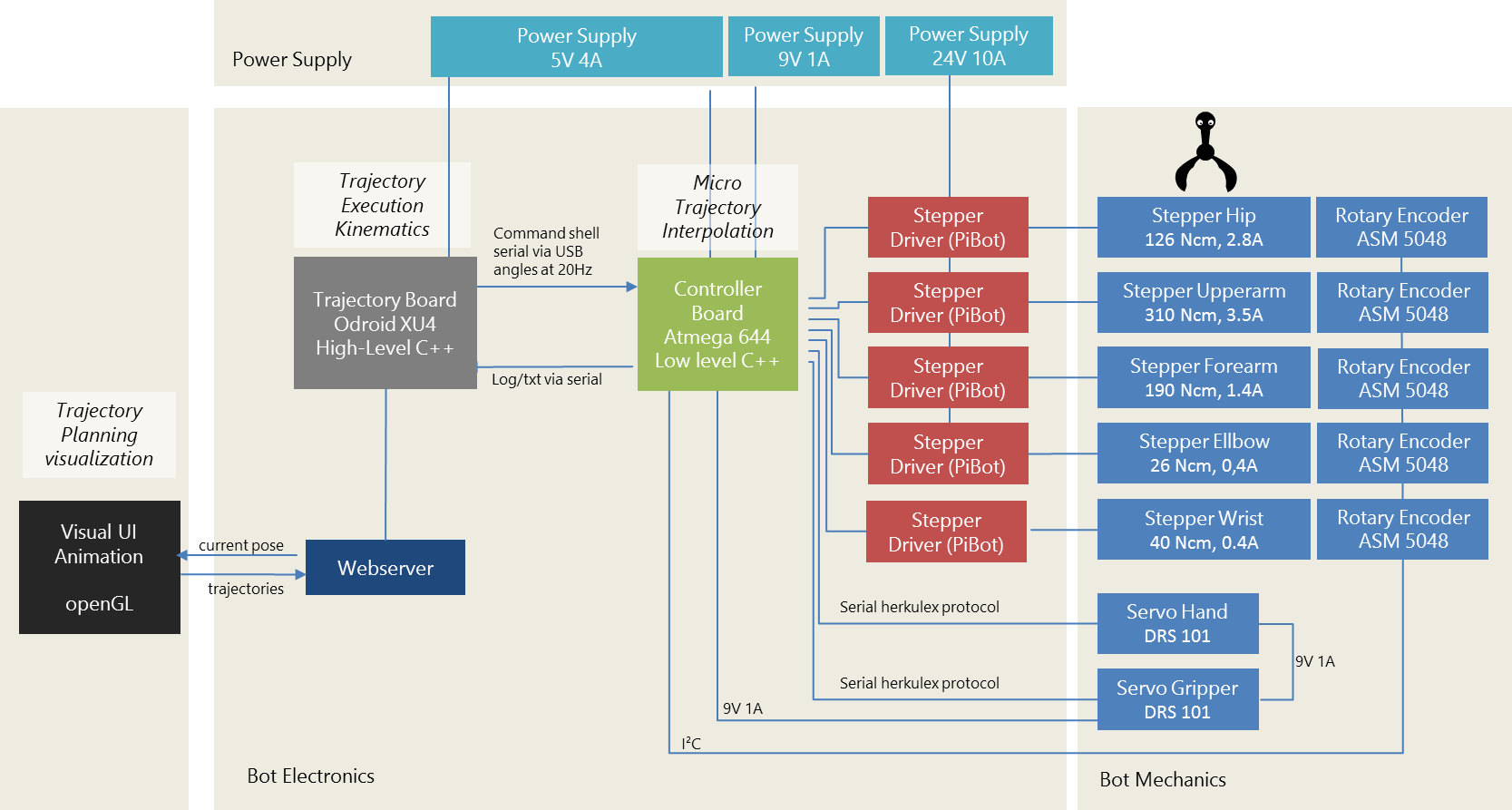


Figure ‑ Architecture

The steppers are driven by retail stepper drivers (PiBot Stepper Driver) around the popular PWM stepper driver Toshiba 6600 providing 4.5A max. The stepper drivers are controlled directly via the Controller Board, which receives joint angles at a sample rate of 20Hz, interpolates in between, and sends the according PWM signal to the stepper drivers and the serial command to the servos. Besides micro interpolation of the trajectory, the controller board takes care of the speed profile, i.e. it limits the acceleration and speed of each actuator. The controller board is a DIY board around an ATMega 644 (for the sheer number of pins) running firmware in low level C++ on base of the Arduino library.

The controller board is fed by the trajectory board, which is an octa core board (Odroid XU4). It runs a defined trajectory (a sequence of poses), interpolates a cubic Bezier curve between the poses and does the inverse kinematics computation. This results in a sequence of joint angles which is sent to the controller board.

The trajectory controller board is encapsulated by a webserver exposing the current movement and accepting commands like new trajectories.

# Kinematics

This chapters explains the kinematics, i.e. the computation of the tool centre point out of joint angles and vice versa. First is demonstrated in chapter 2.1 (simple), latter in chapter 2.2 (rather tricky).

But before starting any kinematics, it is necessary to define all coordinate systems.

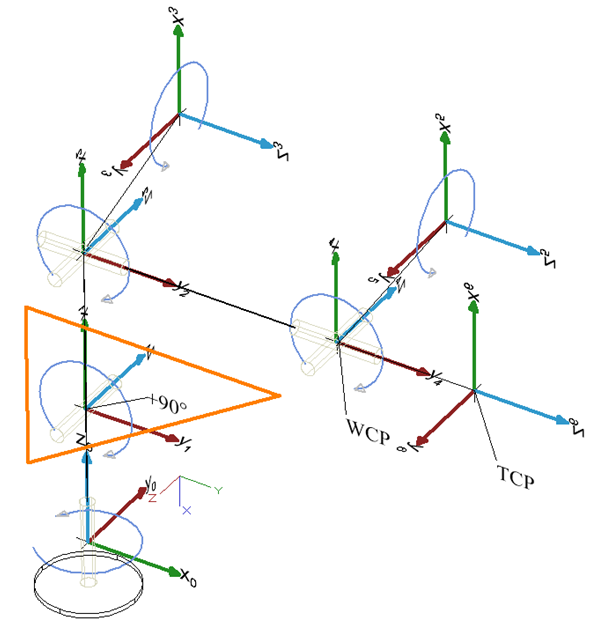


Figure ‑ Coordinate Systems in default position

The picture shows the used coordinate systems in the default position of the bot, having all angles at 0°, starting from the base (angle0) and ending with the coordinate system of the hand (angle6). For convenience the forearm (angle1) adds +90° to the real angle in order to have the base position at 0°of the bot, although the illustrated actually is -90°. The coordinate systems are arranged according to the Denavit Hardenberg convention, which is:

* The angle rotates along the z-axis
* The z-axis points on the direction of the next joint
* The transformation from anglei to anglei+1 is done via

1. rotating around the x-axis by 
2. translation by *a* along the x-axis
3. translation by *d* along the z-axis, and a
4. rotation around the z-axis induced by the joint angle

So, the Denavit Hardenberg parameters are:

|  |  |  |  |
| --- | --- | --- | --- |
| Joint | a[°] | a[mm] | d[mm] |
| hip | -90° | 0 | d0 |
| upperarm | 0 | a1 | 0 |
| forearm | -90° | 0 | 0 |
| ellbow | 90° | 0 | d3 |
| wrist | -90° | 0 | 0 |
| hand | 0 | 0 | d5 |

Table Denavit Hardenberg parameters

The general definition of a Denavit Hardenberg transformation is

|  |  |
| --- | --- |
|  | (2‑1) |

Combined with the DH parameters, the following DH matrixes define the transformation from one joint to its successor:

|  |  |
| --- | --- |
|  | (2‑2) |
|  | (2‑3) |
|  | (2‑4) |
|  | (2‑5) |
|  | (2‑6) |
|  | (2‑7) |

## Forward Kinematics

With the DH transformation matrixes at hand, computation of the bot’s pose out of the joint angles is straight forward. The matrix representing the gripper’s pose is

|  |  |
| --- | --- |
|  | (2‑8) |

By multiplying the transformation matrix with the origin (as homogeneous vector), we get the absolute coordinates of the tool centre point in world coordinate system (i.e. relative to the bot’s base).

|  |  |
| --- | --- |
|  | (2‑9) |

The orientation in terms of roll/nick/yaw of the tool centre point can be derived out of by taking the part representing the rotation matrix (). [[1]](#footnote-1)

|  |  |
| --- | --- |
|  | (2‑10) |
|  | (2‑11) |
|  | (2‑12) |
|  | (2‑13) |

Due to singularities, we need to consider and use

|  |  |
| --- | --- |
|  | (2‑14) |
|  | (2‑15) |

Instead. if

|  |  |
| --- | --- |
|  | (2‑16) |
|  | (2‑17) |

**Note:** Unfortunately, the gripper’s coordinate system is not appropriate for human interaction, since the default position as illustrated in Figure 2‑1 is not. So, it is handy to rotate the gripper matrix such that the default position becomes. The according rotation matrix represents a rotation of -90° along x,y, and z, which results in a simple rotation matrix of

|  |  |
| --- | --- |
|  | (2‑18) |

In the following, this is not considered, but we stop at the coordinate system to simplify the computation.

## Inverse Kinematics

Inverse kinematics denotes the computation of all joint angles out of the tool-centre-point’s position and orientation. In general it is hard to give non-numeric solution, in this case it is possible since the upper three joint angles point to one point, the so-called wrist centre point (Figure 1‑1).

We know the TCP’s position and orientation in terms of roll, nick, yaw (.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | (2‑19) |

First, we need to compute the wrist-centre-point out the tool-centre-point. This is possible by taking the TCP and moving it back along the TCP’s orientation by the hand length. For doing so, we need the transformation matrix from the base to the last joint , which we can derive out of the TCP’s position and orientation.

To build the transformation matrix we need the rotation matrix defining the orientation of the tool-centre-point. This is given by multiplying the rotation matrixes for all axis which gives

|  |  |
| --- | --- |
|  | (2‑20) |

Now we can denote the transformation matrix of TCP

|  |  |
| --- | --- |
|  | (2‑21) |

From the TCP’s perspective, WCP is just

|  |  |
| --- | --- |
| = | (2‑22) |

Furthermore, , so we get the wrist-centre-point by

|  |  |
| --- | --- |
|  | (2‑23) |

1. <https://de.wikipedia.org/wiki/Roll-Nick-Gier-Winkel#Berechnung_aus_Rotationsmatrix> [↑](#footnote-ref-1)