Locomotion Flowchart Explanations – Rev 2

**Bold = function/module**

Regular = data stream

* *Input: Coordinates and orientation to move to and current location + orientation, desired velocity*
* **Trajectory Planner: maps series of steps and turns required to reach desired state - based on current state, current step execution (if updated mid-step), and currently planned path (if updated mid-path)**
  + Step Sequence: List of steps and turns required to reach desired state
* **Trajectory Buffer: Holds currently planned trajectory**
  + Next step: Information about next “planned “ step, based on original planned trajectory
* **Step Buffer: Holds current step(s) to execute and modifies them based on feedback – these steps are the minimum threshold of steps that will be executed and cannot be overwritten**
  + Current step: Data about current step to execute, modified based on feedback – ideally this should correct for any error and make sure once the step is executed the robot ends up as close as possible to the “planned” step.
* **ZMP Generator: Uses step information to find Zero Moment Point for C.M. (Center of Mass)**
  + ZMP coordinates: Required C.M. coordinates to maintain stability
  + Foot coordinates: Coordinates for swing foot placement
* **Linear Inverted Pendulum Model: Calculates the required trajectory for the C.M.**
  + C.M. Trajectory: Stable C.M. trajectory for current step
* **Torso Trajectory Generator: Relates the C.M. trajectory to the torso of the Nao Robot – should be a simple coordinate transformation**
  + Torso Trajectory: Required path for Nao torso to maintain stability
* **Swing Foot Trajectory Generator: Generates path for swing foot based on final foot placement requirements**
  + Swing Foot Trajectory: Path for swing foot
* **Inverse Kinematic Solver: Using trajectory of swing foot and torso, computes required joint angles for Nao robot to execute the motion based on robot geometry**
  + Joint Angles: Matrix of angles for each joint that the actuators need to move to at each time step to maintain stability
* **Abstraction Layer: Layer to help smooth the control signal sent to the robot and attempt to synchronize processor update speed with actuator movement speed**
  + Control signal: The actual signal being sent to the actuators
* **Actuators: Built in Nao actuators**
* **Sensors: Built in Nao sensors**
  + Measured Joint Angles: Actual angles attained by actuators
* **Forward Kinematic Solver: Computes actual position of Nao based on sensor feedback**
  + Foot Position: Actual foot position of Nao as determined by FK Solver
  + C.M. Position: Actual C.M. position of Nao as determined by FK Solver
* **Step Controller: Computes error of C.M. and swing foot on current step**
  + Step error: Deviation of the Nao’s C.M. and swing foot from planned motion

Notes

* What are the inputs (coordinates to move to and current location) based on? Foot coordinates? C.M. coordinates? Something else? Need to talk to Localization people
* Current plan is to have pre-processing done in Lua and everything else in C++
* Might need to include falling conditions somewhere