I. HUBO2 PLUS

Hubo2 Plus, Hubo for short, is a 130 cm (4' 3") tall, 42 kg (93 lb) full-size humanoid robot. It was designed and constructed by Prof Jun-Ho Oh and the Hubo Lab at the Korean Advanced Institute of Science and Technology[1]. Hubo has 2 arms, 2 legs and a head, making it anthropomorphic to a human. It boasts 38 degrees of freedom (DOF) consisting of 6x in each leg, 6x in each arm, 5x in each hand, 3x in the neck, and 1x in the waist. All joints with the exception of the fingers are high gain PD position controlled. The fingers are PWM controlled. It has a three axis force torque (FT) sensor on leg between the end of the ankle and the foot and on the arm where it connects to the hand. Additionally it has accelerometers on each foot and a six axis inertial measurement unit (IMU) slightly below it's weight (approximately the centre of mass). The reference commands for all of the joints are sent from the primary control computer (x86) to the individual motor controllers via two Controller Area Network (CAN) buses. There are currently eight Hubo's functioning in the United States as of December 2012. Four reside at Drexel University and one at Georgia Tech, Perdue, Ohio State and MIT. Jaemi Hubo is the oldest of the Hubos in America and has been at the Drexel Autonomous Systems Lab[1] (DASL) since 2008[2]. Fig. 1 shows the major dimensions of Hubo.

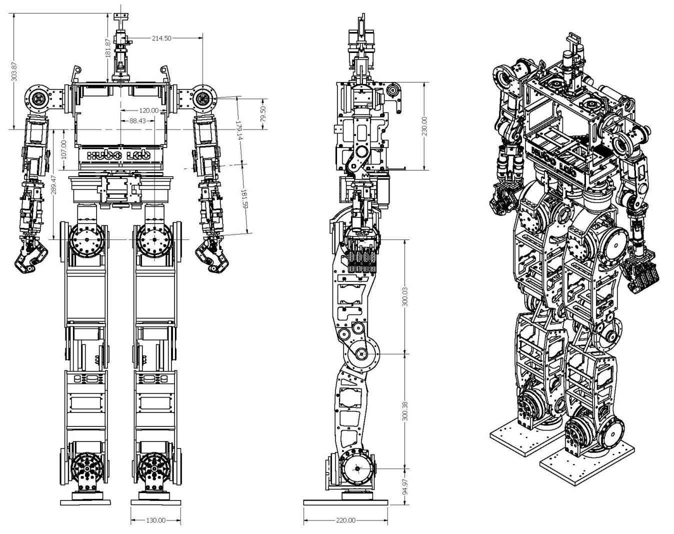

Fig. 1. Hubo2 platform: 130 cm tall full-size humanoid robot weighing 37 kg. It has 38 DOF consisting of 6x in each leg, 6x in each arm, 5x in each hand, 1x in the waist, and 3x in the neck.

II. HUBO-ACH: LINUX ON HUBO

Hubo now runs on the Linux based, open-source, BSD licensed, system called Hubo-Ach. Hubo-Ach is the brain child of Daniel M. Lofaro[2] of DASL at Drexel University in collaboration with *Golems - The Humanoid Robotics Laboratory*[3] at the Georgia Institute of Technology.

The overarching goal of the Hubo-Ach system is to create an easy to use interface between the Hubo hardware and the software environment. All system design decisions are made with the users, programmers and developers of the Hubo in mind. This design philosophy streamlines closed-loop controller implementation, human robot interaction development and the utilization of popular robot related systems such as ROS[4] (Robot Operating System), OpenRAVE[5] and MATLAB[6] on the Hubo platform.

Hubo-Ach is a single process that uses a high-speed, low-latency IPC called Ach [3] to comunicate with controllers that are indipendent processes. Controllers are able to command each joint over an Ach channel at arbratary rates. Hubo-Ach updates the motor references with the latest reference on the feedforward chanel at the rising edge of the real-time loop. The real-time loop in Hubo-Ach is needed to ensure the internal phase lock loop (PLL) of the motor controller lock onto the reference update rate and to ensure the CAN bus bandwidth is not saturated. This loop runs with a period of T_0. T_0 is currently set to 0.005 sec. This causes a CAN bus utiliztion of 78%.

Fig. 2 shows how OpenHUBO (*Process 1*) gets feedback from Hubo via the feedback Ach channel populated via Hubo-Ach. The OpenHUBO requests feedback data with a period of T_1. If $T_1 > T_0$ OpenHUBO will receive the most recent state data received by Hubo-Ach from the Hubo with no overlapping frames. If $T_1 < T_0$ OpenHUBO will receive the latest state data received by Hubo-Ach from the Hubo but some frames will overlap causing identical state data. OpenHUBO performs calculations for tasks such as inverse kinimatcs for picking up cup, foot placement, etc. Reference commands are created for each task and sent to *Process 2* with a update period of T_2. *Process 2* is a low-pass filter that limits the jerk on each joint. The resulting trajectory is sent to the Feedforward Ach chanel with a period of T_3. T_3 is currently set to 0.01 sec. This means that T_2 does not have to be a regular rate, it is possiable for $T_2 >> T3$ and the commands sent to *Process 2* can be a step input. The resulting movements of Hubo are smooth and jitter free.

The key point is that Hubo-Ach updates the state data in the feedback Ach channel commands the motors with the references from the feedforward Ach channel in real-time with a preiod of T_0 no matter what rate the external controller is updating the feedforward channel.

REFERENCES

[1] Baek-Kyu Cho, Sang-Sin Park, and Jun ho Oh. Controllers for running in the humanoid robot, hubo. In *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, dec. 2009.

[2] Daniel M. Lofaro, Robert Ellenberg, Paul Oh, and Jun-Ho Oh. Humanoid throwing: Design of collision-free trajectories with sparse reachable maps. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, oct. 2012.

[3] N. Dantam and M. Stilman. Robust and efficient communication for real-time multi-process robot software. In *International Conference on Humanoid Robots (Humanoids)*, 2012.

[1]Drexel Autonomous Systems Lab: http://dasl.mem.drexel.edu/

[2]Daniel M. Lofaro: http://danlofaro.com/

[3]Golems - The Humanoid Robotics Labatory: www.golems.org/

[4]ROS: http://www.ros.org/

[5]OpenRAVE: http://openrave.org/

[6]MATLAB: http://www.mathworks.com/

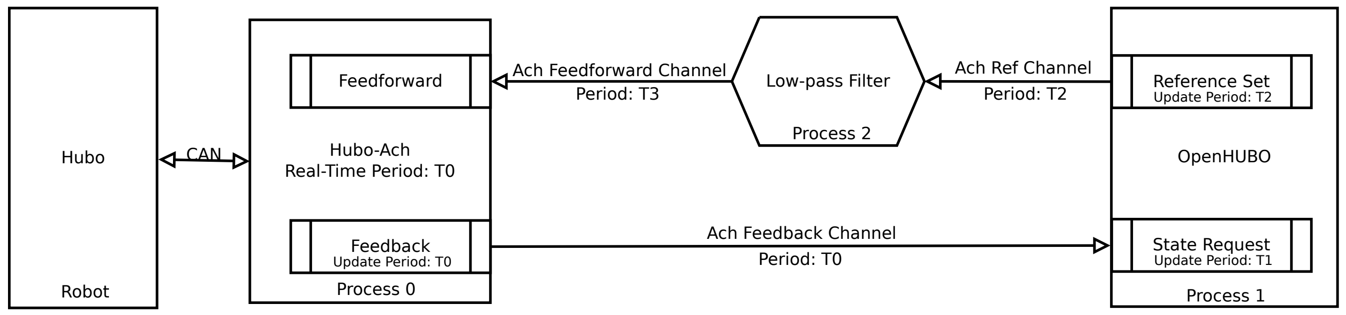

Fig. 2. Hubo-Ach interfacing with two seperate processes creating a closed loop system. Process 0 (Hubo-Ach) takes the most recent command from the Feedforward Ach chanel and sends the reference command over the CAN bus to the Hubo. The state date, including force-torque date from the ankles and wrists, IMU feed back, and actuial joint positions from the encoders are received and stored in the Feedback Ach chanel. Process 1 (OpenHUBO) reads this data at its own rate, proforms the desired task such as calculating the inverse kinimatics inorder to pickup a cup and sends the the desired references to Process 2. This process is a low-pass filter that reduces the jerk on each joint. This smooths the desired trajectory allowing Process 1 to send step input to any of the joints. The reference command for each joint is set on the Feedforward Ach channel at a regular rate compleating the closed loop system.