nature protocols

SUPPLEMENTARY INFORMATION

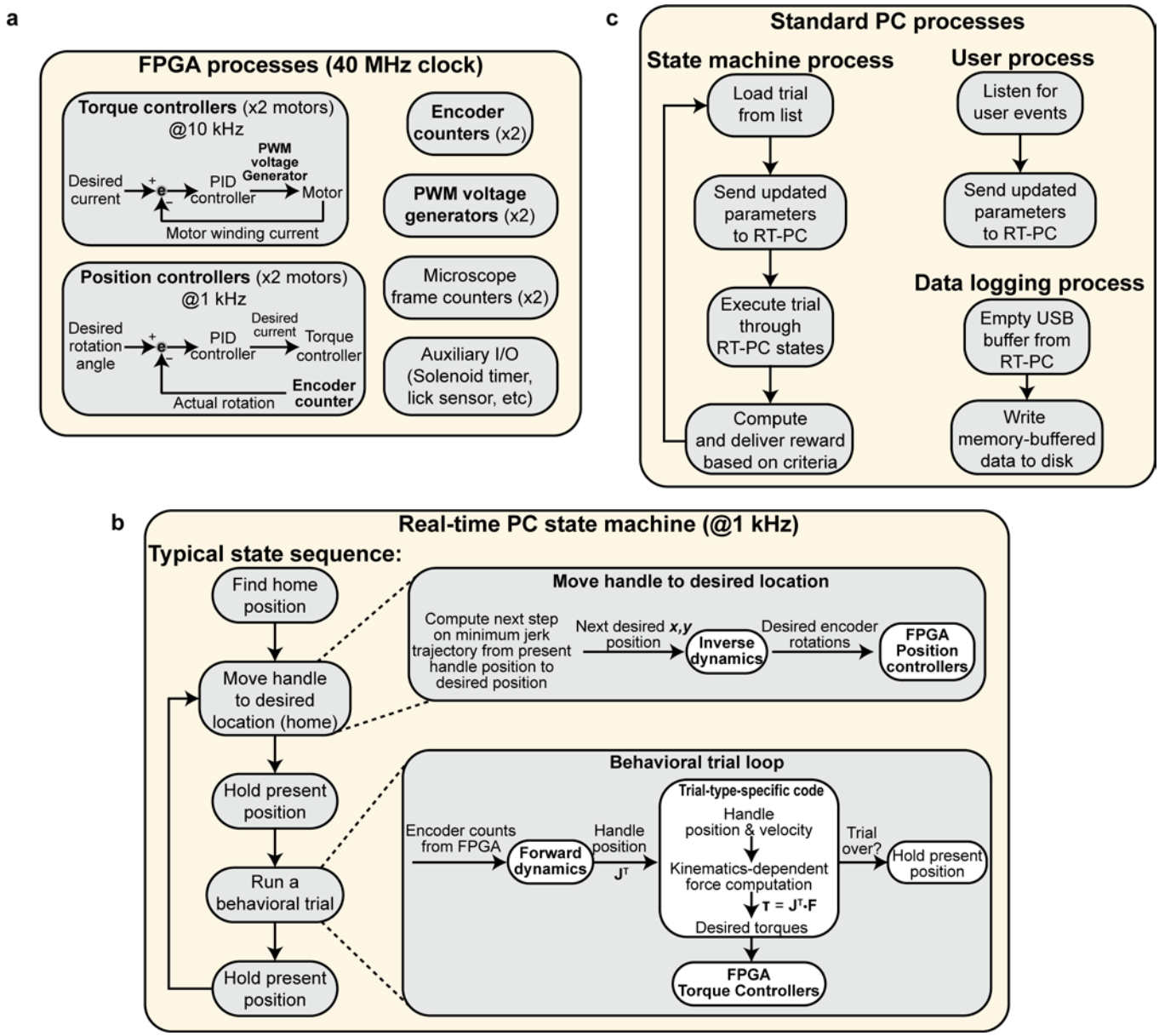
https://doi.org/10.1038/s41596-019-0286-8

In the format provided by the authors and unedited.

# Skilled reaching tasks for head-fixed mice using a robotic manipulandum

**Mark J. Wagner[1*], Joan Savall[1], Tony Hyun Kim[1,2], Mark J. Schnitzer[1,3*] and Liqun Luo[1*]**

[1]Department of Biology and Howard Hughes Medical Institute, Stanford University, Stanford, CA, USA. [2]Department of Electrical Engineering, Stanford University, Stanford, CA, USA. [3]Department of Applied Physics, Stanford University, Stanford, CA, USA. *e-mail: mjwagner@stanford.edu; mschnitz@stanford.edu; lluo@stanford.edu




**Supplementary Figure 1**

Detailed software and controller architecture

**a**, The FPGA uses a PID controller to control motor torques at 10 kHz, and contextually uses an additional PID controller to control motor rotational position at 1 kHz. It also generates variable PWM voltage output to drive the motors, counts digital pulses into an integrated encoder counter, and performs auxiliary device control.

**b,** The real-time PC operates in a state machine. It first locates the home position by incrementing one motor position (via the FPGA position controller) sequentially until the robot physically presses on the outer M3-screw-indexed stop location; it then repeats this process for the other motor and the other outer M3-screw-indexed stop location. It then computes the corresponding “home location” from the measured encoder offsets and the known robot geometry. It then computes a (2 s-long) minimum jerk trajectory[19] from the present position to the home position, and sequentially commands the FPGA position controller with each position in the computed trajectory. It then instructs the FPGA to hold the handle at the home position, and does not begin executing the trial until it senses that the mouse is not pushing on the locked handle (this is assessed by the amount of current that the position controller must produce in order to successfully hold the handle at the home position. If the mouse is not pushing on the handle, very little current will be needed to hold the handle, as there is no external force to compensate). When the trial begins executing, the controller operates a loop in which

it first uses the forward kinematics to compute the handle cartesian position, and then executes a specific code block depending on the present trial type.

For constrained-trajectory-type trials (e.g., Fig. 4b,c), the robot senses the mouse's instantaneous deviations from the constrained trajectory and attempts to cancel out the movement that is orthogonal to the track orientation using a PID controller. For aiming movements, no force is applied unless the user has chosen to apply simulated friction ("viscosity" parameter). Finally, when the movement reaches a defined endpoint—8 mm radial distance in our examples trial list files—it ends the trial by instructing the FPGA position controller to hold the present position. The enumerated states as written to the data stream are: 0: Hold position; 1: Movement; 2: Homing; 3: Go to target; 4: Go to home position; 5: Disabled, which are defined in "Robot state.ctl" (Fig. 5b).

**c,** The standard PC consists of three main processes. The first is a high-level state machine that loads a trial from the user-selected trial list, updates relevant trial parameters accordingly, including sending updated parameters to the RT-PC over the USB stream. The state machine then instructs the RT-PC to begin a trial, waits for the animal to begin movement, terminates the trial if a user-programmed timeout period is exceeded, and otherwise waits for the RT-PC to signal trial completion. Next, the state machine determines the quantity of reward to deliver to the animal based on the user-selected criteria, and instructs the FPGA to open the solenoid for the corresponding duration. The second standard PC process is the front panel user interface event handler, which listens for user-entered data on the interface, and updates parameters both internally and on the RT-PC via USB as appropriate. Finally, a data logging process periodically polls the USB stream for new data from the RT-PC, empties the USB buffer into a memory buffer on the standard PC, and at a time of the user's choosing will empty the memory buffer into a user-selected data file