Lab 6: Virtual Worlds with Haptic Feedback

Lab 6: Virtual Worlds

- We now have everything we need to build virtual systems:
  - Input (quadrature decode using the eTPU)
  - Output (PWM motor control using the eMIOS)
  - Time (interrupt processing)
- Virtual Wall-Damper
  - Add damping to the virtual wall to reduce “chatter”
- Virtual Spring-Damper
- Virtual Spring-Mass
- Virtual Spring-Mass-Damper
- Virtual Knob

Lab 6: Virtual Worlds

- Virtual Spring-Damper
  - Similar to virtual spring (Lab 4) except puck is now connected to the reference point with a damper as well as a spring.
  - Motor must supply torque equal to the sum of the spring and damper torques.
  - Need velocity to compute damper torque. How?
- Virtual Spring-Mass and Spring-Mass-Damper
  - Implement the Spring-Mass difference equations using Forward Euler
  - Stability?
  - How much damping is required?

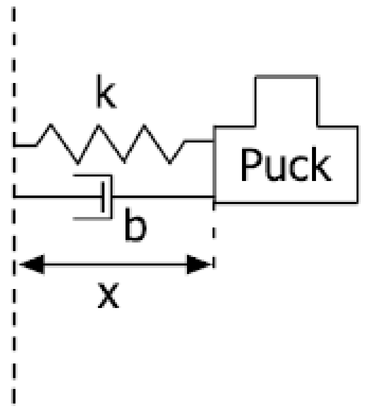

Figure 1: Spring-Damper System

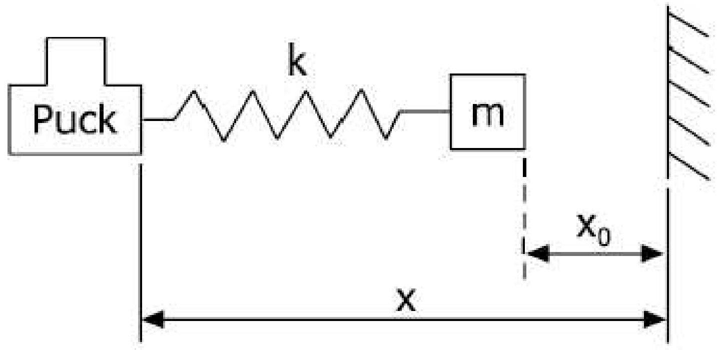

Figure 2: Spring-Mass System

Lab 6: Virtual Worlds

- Virtual Knob
  - See the "TouchSense" product document on the web
  - Programmable tactile feedback used for video editing, medical devices, control rooms
    - "intuitive and precise control"
  - Library of tactile effects
    - Detent
    - Spring
    - Barrier
    - Vibration

Automotive
Programmable rotary modules are used for driver primary controls as well as secondary controls such as climate, media, navigation, and other functions. Controls using programmable tactile feedback can reduce driver glance time. In addition to a potential safety improvement, touch contributes to the multimodal experience of higher perceived quality.

Lab 6: Knob with detents at regular intervals (stereo receiver, for example)

Lab 6: Software

- Use previously developed FQD, PWM, and ISR code for I/O
- `worlds.h` and `worlds.c` compute the torque for the virtual worlds based on the current angle and velocity
  - `float virtualSpringDamper(float angle, float velocity);`
  - `float virtualWallDamper(float angle, float velocity);`
  - `float virtualSpringMass(float angle);`
  - `float virtualSpringMassDamper(float angle, float velocity);`
  - `float virtualKnob(float angle, float velocity);`
- `Lab.c` establishes a periodic interrupt using the DEC
- One ISR for each virtual world calculates velocity and outputs torque command to the motor

  -
    ```
    void sdIsr(void) {
    /* ISR for the spring-damper system */
    /* Calculate the wheel's velocity */
    /* Calculate and apply torque to haptic wheel */
    ```

Lab 6: Model-based Systems Engineering

- For the spring-mass-damper system, we want to choose values of k, m to provide specified properties
- Verification model (continuous time)
  - Oscillator with frequency = 1Hz, maximum torque = 800 Nmm
  - Verify your design in simulation *before* you implement
- Verification model (discrete time)
  - Unstable due to Forward Euler integration
  - Add damping (how much?) to create a discrete time harmonic oscillator
  - Verify in simulation before you implement
  - Eventually, generate code directly from the discrete time verification model!