Spring 2021 RTOS Project

Inverted Pendulum game

Version 0.1

March 22, 2021

Authored by: Jon Haines

Instructions on how to earn credit for delivering the right work product for this project during this semester can be found on Canvas.

Basic idea:

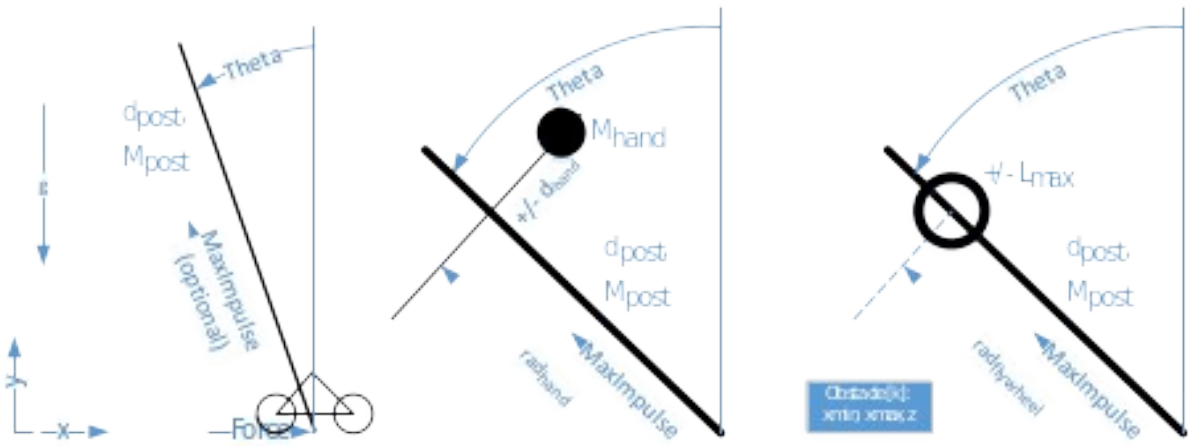
To model the physics behind a simple mechanics problem, with some environmental inputs that will affect the physics, with inputs from our SDK affecting the system and with the physical state represented in graphics and LEDs on our SDK.

> https://en.wikipedia.org/wiki/Inverted_pendulum
>
> https://en.wikipedia.org/wiki/Pogo_stick

One of the great advantages to our use of the SDK to simulate this is that we can have ideal conditions (no slippage where we don't want it, no friction anywhere else; motors that have instant response and constant torque when we want it; configurable gravity; effectively-infinite K springs; etc).

Variants suggested for consideration:

1. Inverted pendulum (first figure, with MaxImpulse=0)
2. Inverted Pendulum with impulse kicker (first figure, with MaxImpulse>0)
3. Pogo Stick (second figure)
4. Pogo Stick with flywheel (third figure)



Baseline project expectations (if you want to get buy-in for your own idea, create 6 analogous elements):

- Physics that gets serviced every $\tau_{phy.}$ (Part of your work in a later week on your project is going to be to push up the physics update frequency to find out when you no longer have uniform physics)
    - The Impulsive Inverted Pendulum is going to be a little weird. It will have magical impulse given to it (simply add momentum to the pendulum at the time of impulse, in the post's direction), and upon landing will conserve the translational kinetic energy of the system, but has a perfectly inelastic collision in the y direction (adjust x velocity to maintain x momentum, and the y velocity at the bottom but not the top).
    - The pogo sticks have perfectly static friction between the earth and the post's bottom end.
- CapSense Slider
    - Is used to adjust:
        - Force being applied to the mobile base (Variants 1,2)
        - Offset position of the mass on the "handlebar" of the pogo stick (Variant 3)
        - The flywheel's angular momentum (Variant 4)
    - May be used as a 4-position slider, or you can use the interpolating feature of
- Pushbutton controls and first LED
    - For the simple inverted pendulum:
        - The buttons turn down and up the gain for the Force on the bottom of the pendulum, per button push. The first LED is pulse width modulated to show the gain.
    - For all Impulse Kicker systems:
        - First button is used to pre-load the kicker based on how long the button is held between kicks, second button kicks. First LED is pulse width modulated to show how much the kicker is pre-loaded
- Second LED
    - For the simple inverted pendulum, and pogos without obstacles: light and stay lit until game is reset if Theta ever reaches +/- π/2. (The post has hit the ground)
    - For pogos with obstacles: light and stay lit until game is reset if the base of the post ever is observed to through the boundaries of an obstacle (pogo has been tripped up by not clearing the obstacle), OR if the top end of the pogo hits the ground.
    - If neither of those conditions occur but the base of the pole leaves the graphed xmin,xmax range, the second LED shall continue flashing with a 1Hz frequency until the game is reset.
- Graphics to show the pendulum's post, the ground, and indication of rotation/translation. You must use the bottom of your display as the "ground", but don't need to draw a line there. The picture should update every $\tau_{display}$ (As with the physics update, you'll be finding out how fast you can push your solution in later weeks.) Figure out the pixel geometry of your SDK's LCD screen, and make the xmin and xmax (see below) relate to the outer edges of the leftmost and rightmost pixels of your screen, and scale the y axis assuming that the pixels are physically square.
- Configurable data to drive the game, with clear and appropriate units noted in comments for each 32b quantity shown below. (Your data will have space for each of these, even if you don't implement anything for that datum.)
    - Data Structure Version (Defined as 1 for this table)
    - Gravity [mm/s^2]
    - Mass of Post [g]

- Length of Post (d_{post} in picture above) [mm]
- Graphing Limits
- (signed) Xmin [mm]
- (signed) Xmax [mm]
- Variant selected [enumeration, per Variant list]
- MaxForce (for Variants 1,2) [mN]
- MaxImpulse (for Variants 2-4) [g-m/s]
- Non-pole Mass (for Variants 3,4)
    - Mass [g]
    - Center-of-mass distance from bottom of pole [mm]
- Maximum distance from pole for the additional mass to move (Variant 3) [mm]
- Maximum Angular Momentum (Variant 4) [g-mm^2/s]
    - Note: the current angular momentum is “instantly adjustable” via your slider, as if the flywheel has infinite angular acceleration for an infinitesimal time.
- Number of Obstacles
- Obstacle[x Number of Obstacles]:
    - (signed) xmin [mm]
    - (signed) xmax [mm]
    - height [mm]