Introduction:
This project is a cyber-physical model of a subsystem from the bartending robot I am building for my senior design project. This cyber-physical model describes the behavior of liquid inside of a valved container which is controlled by a microcontroller. This system receives commands from the main controller to either dispense liquid or return the liquid level sensor value. The main controller should not send dispense commands if the liquid level in the container is below an operational level but the model accounts for this behavior anyway in the interest of robustness.

Purpose:
The purpose of this project is to design a closed loop feedback controller that controls the valve timing of a container to dispense any given volume of liquid. Feedback from a liquid level sensor is required because the flow rate out of the container is non-linearly related to the height of the liquid in the container. Due to this nature the valve timing must be calculated every single time, which can be done with a model of the physical system.

Cyber-Physical:
The design and modeling in this project features cyber and physical components that are interconnected to achieve full control of the system. The functionality of this system depends on a relationship between the cyber and physical components. Without the sensor measuring the height of the liquid in the container, the controller would not be able to determine the valve timing nor provide information about the status of the container. Without the cyber components the valved container does nothing unless human actions are involved, making it just a regular container. The components connected in this system allows for very accurate container control which performs exactly as designed. Even having a human attempt to pour a drink from the container by turning the valve on or off would not be nearly as accurate (consistently) as a properly designed cyber-physical system.

Modeling

Information:

Physical:

$V_s$: Valve status ; $V_s = 1$ when valve open, $V_s = 0$ when valve is closed
$H_s$: Current liquid height sensor value
$h$: Current liquid height
$A_s$: Surface area of the bottom of the container
$A_p$: Area of the valve port
$C_d$: Coefficient of discharge ($C_d = C_v$ (velocity coefficient) $\times C_c$ (contraction coefficient))
$g$: Gravity ($9.81 \text{ m/s}^2$)

Cyber:


Inputs:
$I_v$: Input Volume of liquid to dispense
$I_f$: Input Flag; $I_f = 1$ when new input is ready, $I_f = 0$ when all inputs have been read
$I_r$: Input Request; $I_r = 1$ when dispense request, $I_r = 0$ when update sensor request

**Outputs:**

$P_v$: Power Valve: $P_v = 1$ when valve is open, $P_v = 0$ when valve is closed
$H_{out}$: Liquid Height: Last sensor value read
$W_{lv}$: Low Volume Warning: $W_{lv} = 1$ when liquid height is too low, $W_{lv} = 0$ when liquid height is in operational range

**Components and Variables:**

$\tau_w$: Work timer while performing tasks
$\Delta V$: Last recorded dispense volume input
$H$: Liquid height value measured
$T_s$: Sensor value read time
$T_c$: Calculation time
$T_d$: Dispense time
$T_{ws}$: Time interval between sensor readings in the warning state

**Math:**

**Physical**

\[
\begin{align*}
\dot{h} &= -\frac{V_s C_d A_s \sqrt{2g}}{A_s} \\
V_s &= P_v \\
H_s &= h
\end{align*}
\]

**Cyber**

\[
\begin{align*}
\dot{q} &= 0 \quad \text{when } q = \text{waiting} \\
\dot{\tau}_w &= 0 \\
q^+ &= \text{sensing} \\
\tau_w^+ &= T_s \quad \text{when } q = \text{waiting} \text{ and } I_r \geq 1 \text{ and } H > H_{min} \\
\Delta V^+ &= I_v \\
\dot{q} &= 0 \\
\dot{\tau}_w &= -1 \quad \text{when } q = \text{waiting} \text{ or } q = \text{calculating} \text{ or } q = \text{dispensing} \text{ or } q = \text{lowvolume} \\
P_v &= 0 \\
q^+ &= \text{lowvolume} \\
H^+ &= H_s \quad \text{when } q = \text{sensing} \text{ and } H \leq H_{min} \text{ and } \tau_w \leq 0 \\
H_{out} &= H \\
\tau_w^+ &= T_{ws} \\
q^+ &= \text{waiting} \\
H^+ &= H_s \quad \text{when } q = \text{sensing} \text{ and } I_r = 0 \text{ and } H > H_{min} \text{ and } \tau_w \leq 0 \\
H_{out} &= H
\end{align*}
\]
\[
\begin{array}{l}
q^+ = \text{calculating} \\
H^+ = H_s \quad \text{when } q = \text{sensing and } I_r >= 1 \text{ and } H > H_{min} \text{ and } \tau_w \leq 0 \\
H_{out} = H \\
\tau_w^+ = T_{ws} \\
q^+ = \text{dispensing} \\
\tau_w^+ = \frac{A_0(2\sqrt{\pi} - 2\sqrt{\frac{H-y}{2\rho g}})}{C_d A_a \sqrt{2g}} \quad \text{when } q = \text{calculating and } \tau_w \leq 0 \\
P^+_v = 1 \\
q^+ = \text{waiting} \quad \text{when } q = \text{dispensing and } \tau_w \leq 0 \\
P^+_v = 1 \\
\end{array}
\]

Physical Model Breakdown:
To model the flow rate out of the container we must first perform some basic intermediate calculations. Initially, the velocity of the fluid out of a spigot must be calculated using Bernoulli’s equation:

\[
P_1 + \frac{\rho v_1^2}{2} + \rho g y_1 = P_2 + \frac{\rho v_2^2}{2} + \rho g y_2
\]

Where \( P \) is the system pressure, \( \rho \) is the fluid density, \( y_2 \) is the fluid height, \( y_1 \) is the spigot height and \( v \) is the fluid velocity. This model is for a system on earth therefore \( P_1 = P_2 = \) atmospheric pressure. Assuming that the surface area of the liquid in the container is much greater than the surface area of the spigot, the velocity of the fluid in the container will be very close to zero. The model now becomes:

\[
\frac{\rho v_1^2}{2} + \rho g y_1 = \rho g y_2 \rightarrow v = \sqrt{2g(y_2 - y_1)}
\]

The general model for volume flow through an aperture is given as:

\[
V = C_d A_a v
\]

where \( V \) is the volume flow rate, \( v \) is the velocity of the fluid flowing through the aperture, \( A_a \) is the area of the aperture, and \( C_d \) is the discharge coefficient. The discharge coefficient is equal to the velocity coefficient \( (C_v) \) multiplied by the contraction coefficient \( (C_c) \). The model for the volume flow through an aperture is now:

\[
V = C_d A_a \sqrt{2g(y_2 - y_1)}
\]

Assuming that the height of the aperture is approximately zero, the model can be simplified to:

\[
V = C_d A_a \sqrt{2gh}
\]

Calculating the volume that has left the container:

\[
L(t) = C_d A_a \int_0^t \sqrt{2gh(\tau)}d\tau
\]

For this particular application, the flow rate with respect to time is desired therefore we must derive an equation for the height of the container as a function of time:
\[ h(t) = h_0 - \frac{L(t)}{wl} = h_0 - \frac{C_d A_s}{wl} \int_0^t \sqrt{2gh(\tau)} \, d\tau \]

where \( w \) is the width of the container, \( l \) is the length of the container, and \( h_0 \) is the initial height. Now we can take the derivative:

\[ \dot{h}(t) = -\frac{C_d A_s \sqrt{2g}}{wl} \sqrt{h(t)} = -C \sqrt{h(t)} \]

where \( C = \frac{C_d A_s \sqrt{2g}}{wl} \). The general solution to this system is then:

\[ h(t) = \frac{(Ct - 2\sqrt{h_0})^2}{4} \]

Now the volume flow rate out of the container can be written as a function of time:

\[ V(t) = C_d A_a \sqrt{g} (Ct - 2\sqrt{h_0}) \]

Now we can calculate the amount dispensed:

\[ \Delta V = - \int_0^t V(t) \, d\tau = - \frac{C_d A_a \sqrt{2g}}{4} (Ct^2 - 4t \sqrt{h_0}) \]

Solving this equation for \( t \) yields the valve time necessary to dispense given quantity \( \Delta V \)

\[ t = \frac{wl(2\sqrt{H} - 2\sqrt{H - \frac{\Delta V}{A_s}})}{C_d A_a \sqrt{2g}} = \frac{A_s(2\sqrt{H} - 2\sqrt{H - \frac{\Delta V}{A_s}})}{C_d A_a \sqrt{2g}} \]

where \( A_s \) is the surface area of the bottom of the container: \( A_s = wl \)

**Importance of Invariance:**

Having liquids near electronics is unsafe, potentially destructive, and should be avoided whenever possible. My project is composed of electronics that are used to dispense liquid (which is located above the electronics because it a gravity feed system) so complete control of the system is essential. The property of invariance is important to this control system because it is only operational for a specific liquid height range. If the liquid level is within the operational range when the robot initializes, it needs to be able to keep it there indefinitely. The controller design has built in limitations that are designed to keep the liquid level within this range at all times, when the liquid level drops to a certain level the controller issues a refill warning and will not attempt to dispense. If the input dispense volume is beyond the remaining volume of the container, the software would have an error if there weren’t a guard condition. Due to the physics of the model, attempting to dispense a volume greater than the remaining liquid volume in the container results in a calculation with the square root of a negative number. The calculated time from this input would result in an error or an incorrect value depending on the computer architecture being used. The filling of the container is done by a person operating the robot so an overflow is out of the robot’s control. If this were to happen though, the robot should still be able to operate properly unless the circuitry is damaged because the excess liquid will pour out of the top of the container resulting in the maximum working liquid height. If the container was emptied while the robot off, it will send a refill warning to the main controller upon initialization and will not attempt to pour until the container is filled to at least the minimum volume required for dispensing.

**Simulation:**


I was able to implement most of this model in Simulink, and I am very happy with the results. I used the hybrid toolbox to model the physical system controlled by a state machine implemented in the Stateflow toolbox. The only thing I did not implement in Simulink was the user input because I had already implemented it in the controller I had designed. I actually built this system but wasn’t able to get it fully functioning due to the ultrasonic sensor being used for sensing the height of the liquid. I was getting some strange results (although it worked great when sensor input was replaced by manually entering in calibration values that I had marked on the container) and did not want to spend time attempting to calibrate it when I will most likely be designing a different container for my project anyway. Instead, I had the user input be a constant volume with an input flag triggered when a jump occurred following a successful dispense. I realized that it didn’t matter because it was exactly what I would enter if I did have control of the input, and worked great to display exactly the result I wanted to see. Every consecutive dispense takes slightly longer than the previous until the end where each dispense is significantly longer. This is the result described by the model, and also the result I have witnessed in real life before this control system was built.
Bigger Picture: Here is the block diagram of my senior design project to get an idea of where this model fits in.