Stewart Platform Dynamics

Yuwei Wu, Xinyue Wei, Jiatong Sun

Step 1:

Use the simulator to decide poses we want to have.

As long as the length ranges from 0 to 1, it's a safe position

Note: Change line 134 in Platform.pde to the following to mark labels on the actuators

```
134       text(String.format("%d, %.2f",i+1, (l[i].mag() - ACTUATOR_MIN_LENGTH) / ACTUATOR_STROKE_LENGTH), 2,0,0);
```

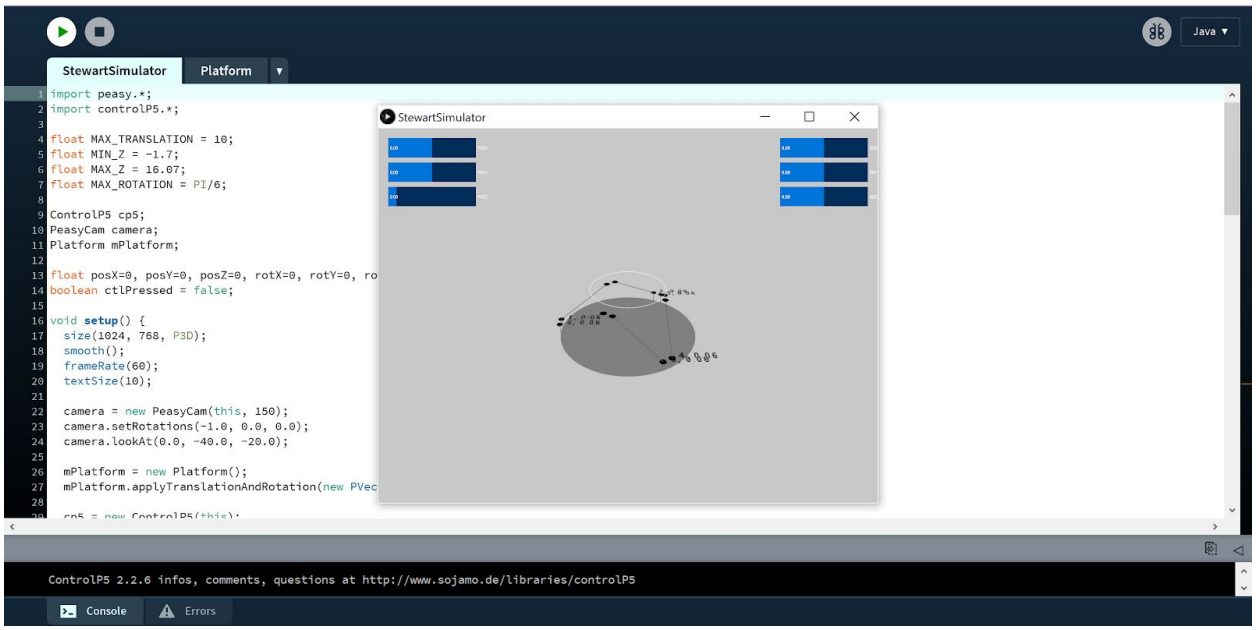

Fig. 1 Simulator

Step 2:

For each leg, we can use any velocity functions to interpolate between two continuous poses. As long as the sum of the extending time equals the sum of the retracting time, the total distance (integral of velocity) of this leg will be equal to zero, which means the leg can go back to its original length after every loop. If we do so for all six legs, we can have the platform back to its original position after every loop.

Math proof:

Assume $x(t)$ is a specific function on interval $[0, t_0]$, with its maximum amplitude as X_m.

Assume $x_1(t)$ and $x_2(t)$ derived from the horizontal stretch of $x(t)$, where $t_1 \in (0, t_1)$ and $t_2 \in (0, t_2)$, or we can write as $x_1(t) = x(\frac{t_0}{t_1}t)$ and $x_2(t) = x(\frac{t_0}{t_2}t)$. Note that $x_1(t)$ and $x_2(t)$ is only compressed horizontally (along t axis) and thus, their maximum value is still X_m.

Assume that $t_1 + t_2 = t_0$, and we can integrate as following:

$$
\begin{aligned}
&\int_0^{t_1} x_1(t)dt + \int_0^{t_2} x_2(t)dt \\
&= \int_0^{t_1} x(\frac{t_0}{t_1}t)dt + \int_0^{t_2} x(\frac{t_0}{t_2}t)dt \\
&= \frac{t_1}{t_0}\int_0^{t_1} x(\frac{t_0}{t_1}t)d(\frac{t_0}{t_1}t) + \frac{t_2}{t_0}\int_0^{t_2} x(\frac{t_0}{t_2}t)d(\frac{t_0}{t_2}t) \\
&= \frac{t_1}{t_0}\int_0^{t} x(t)dt + \frac{t_2}{t_0}\int_0^{t} x(t)dt \\
&= \frac{t_1+t_2}{t_0}\int_0^{t} x(t)dt \\
&= \int_0^{t} x(t)dt
\end{aligned}
$$

So we can use whatever pwm(velocity) functions to interpolate between different poses. As long as the sum of extending time equals the sum of retracting time, the extending distance will equal the retracting distance, so the total distance is zero and the leg can return its original length. In the previous code, it was the parabolic function that was chosen and the time stamp satisfies the requirement of same extending time and retracting time.

```
// amount of time to loop through
int travelTime = 30;
// controls and switches extend/retract for each actuator based on time, starts on extend
int actuatorMovements[6][NUM_MAX_SWITCHES] = {
    {0, 6, 21},
    {0, 6, 21},
    {5, 15, 25},
    {6, 14, 23},
    {10, 25, 30},
    {9, 22, 28}
};
```

Fig. 2 Current Timestamp

```
//2. calculate constants
//what we have: vertex and start/end points, current time, range of y (15-245)
float h = (float(start_time) + float(end_time)) / 2;
float k = 245;

// a = (y - k) / (x - h)^2
// y = y, x = x, h = x vertex, k = y vertex
// point = (start_time, 15)
float a = (15 - k) / (sq(start_time - h));

//3. calculate speed
//y = a(x - h)^2 + k
pwm = a*(sq(t_now - h)) + k;
```

Fig. 3 Current Parabolic Function

Step 3:
The lengths of six legs can determine the pose of the platform.

Since the platform can be regarded as a rigid body, it has 6 degrees of freedom and thus, its pose can be determined by the six lengths of legs.

Summary:
Use the simulator to generate a series of poses we're interested in, obtain six lengths for each pose from the simulator, and since the six lengths can determine the pose, we actually get six series of distances. Decide a certain pwm/speed function to interpolate between two neighboring distances and calculate the timestamp. Implement this timestamp into the Arduino code to generate a new movement loop.