AME 40453: Final Project
Technical Memo
Date Submitted: April 28 2022
Dates Performed: *Spring 2022*
To: Prof. Rumbach
From: Paul Mellor
Subject: Final Project Report

For the Automation and Controls Final Design Project, my project was to build a robotic dog. I had three goals for this project. My first goal was that the dog would be able to walk forward & backward and turn left & right on flat surfaces. My second goal was for the robot to actively balance, so that it did not fall over trying to walk. My third goal was to be able to drive the robot; I wanted to be able to send movement commands and have the dog move accordingly.

To attack these goals, I came up with the design below.

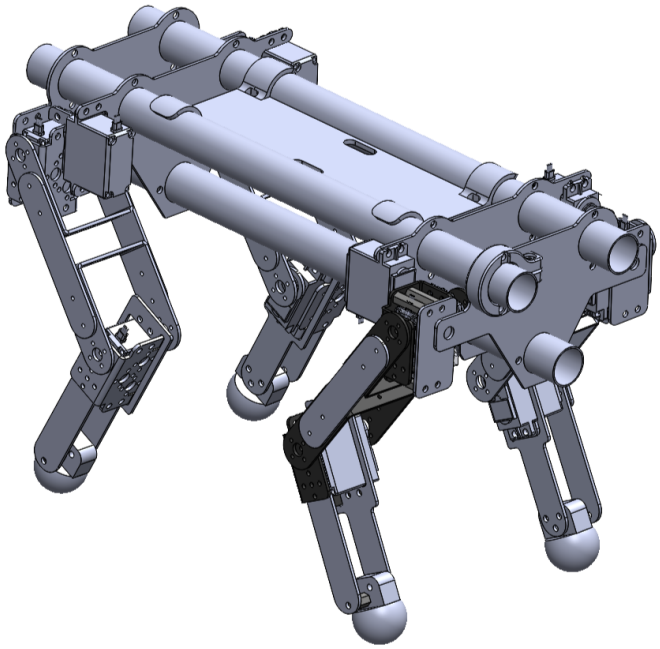

Figure 1: CAD Model of the Robot

The dog is symmetrical, with 4 identical legs having the knee joint facing the center of the dog. Each leg has a shoulder joint with 2 DOFs (degrees of freedom) so that the leg can move forward & backward and left & right. Each joint is driven by a servo motor. Servos are both high torque, and can be positioned accurately. The construction consists primarily of off the shelf brackets, and

2D fabricated plates. Symmetry and reliance on 2D components greatly simplified the fabrication process (most components were cut on a waterjet). Additionally, symmetry allowed any kinematic models for one leg to be applicable to all legs.

The robot's motion is based on positioning the legs. To walk, each leg steps through 6 way-points (See Figure 2), and has a specified time to get from 1 point to the next.

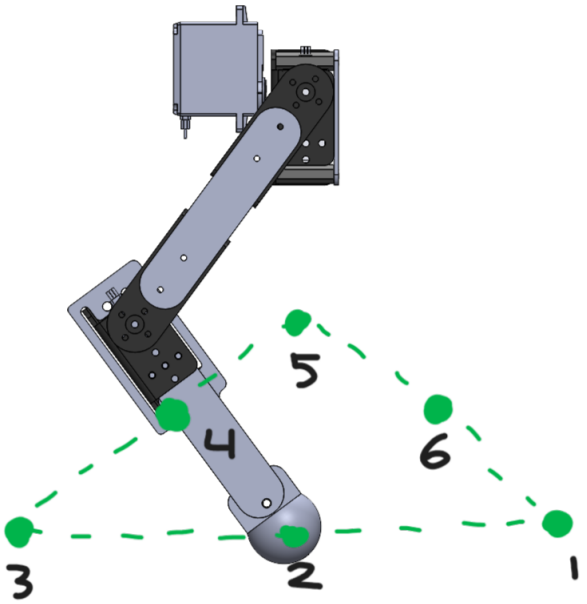

Figure 2: Path/Points of the Leg's Walking Motion

To walk, the legs diagonal from one another move in sync, while the opposite two legs are 180° out of phase (i.e. if one set of diagonal legs is at point 1, the other set is at point 4). The points are all timed so that the leg spends half the time on the ground (points 1, 2, 3), and half the time off the ground (points 4, 5, 6).

Since my approach to moving the legs was to position them in xyz coordinates, I needed an inverse kinematics model to get the motor angles as a function of xyz coordinates for each leg. The inverse kinematics was solved using trigonometry (see Figure 3). This was possible because there are only 3 DOF, and because there is a limited area in which the leg can actually move.

To solve for $\theta_{shoulder1}$ (the shoulder joint moving in the yz plane) using xyz coordinates (as depicted in figure 2) the equation below is used.

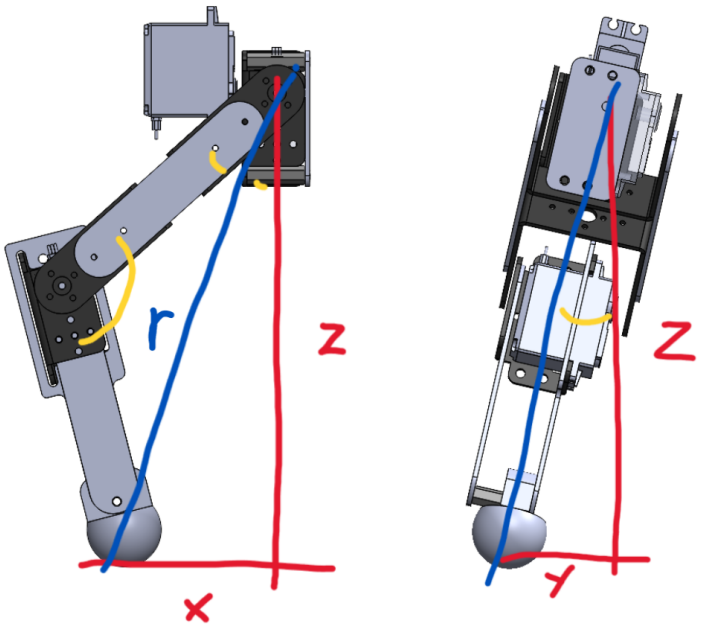

Figure 3: Trig Used to Calculate Inverse Kinematics

$$\theta_{shoulder1} = \arctan(\frac{y}{z}) \tag{1}$$

To solve for $\theta_{shoulder2}$ (the shoulder joint moving in the xz plane) and θ_{knee} an intermediate length r is calculated by:

$$r = \sqrt{z^2 + x^2} \tag{2}$$

This enabled my to use law of cosines to solve for $\theta_{shoulder2}$ and θ_{knee} below.

$$\theta_{shoulder2} = \arccos(\frac{r^2}{2Lr}) - \arctan(\frac{x}{z}); \tag{3}$$

$$\theta_{knee} = \arccos(\frac{2L^2 - r^2}{2L^2}); \tag{4}$$

where L is the length of the limbs (both limbs are equal length).

To actively balance the robot, a controller was designed to maintain $0^\circ$of pitch and roll by keeping

the 4 feet of the robot centered below the COM (center of mass). An accelerometer/gyroscopic sensor was placed on the body of the robot, and returned θ_{pitch} and θ_{roll} of the robot. To control the pitch, the x position of the legs were adjusted using the PID control equation below:

$$x = x_{openloop} + h\sin\left(k_p(\theta_{pitch_s} - \theta_{pitch}) + k_I \int (\theta_{pitch_s} - \theta_{pitch})dt + k_d \Delta\theta_{pitch}\right) \tag{5}$$

where $x_{openloop}$ is the x value before the balance calculation, h is the height of the shoulder joint to the ground, θ_{pitch_s} is the target pitch angle (0°) and k_p, k_I and k_d are the PID control constants. SImilarly, to control the roll, the y position of the legs were adjusted using the PID control equation below:

$$y = y_{openloop} + h\sin\left(k_p(\theta_{roll_s} - \theta_{roll}) + k_I \int (\theta_{roll_s} - \theta_{roll})dt + k_d \Delta\theta_{roll}\right) \tag{6}$$

here $y_{openloop}$ is the x value before the balance calculation, h is the height of the shoulder joint to the ground, θ_{pitch_s} is the target pitch angle (0°) and k_p, k_I and k_d are different PID control constants. The resulting robot is shown below.

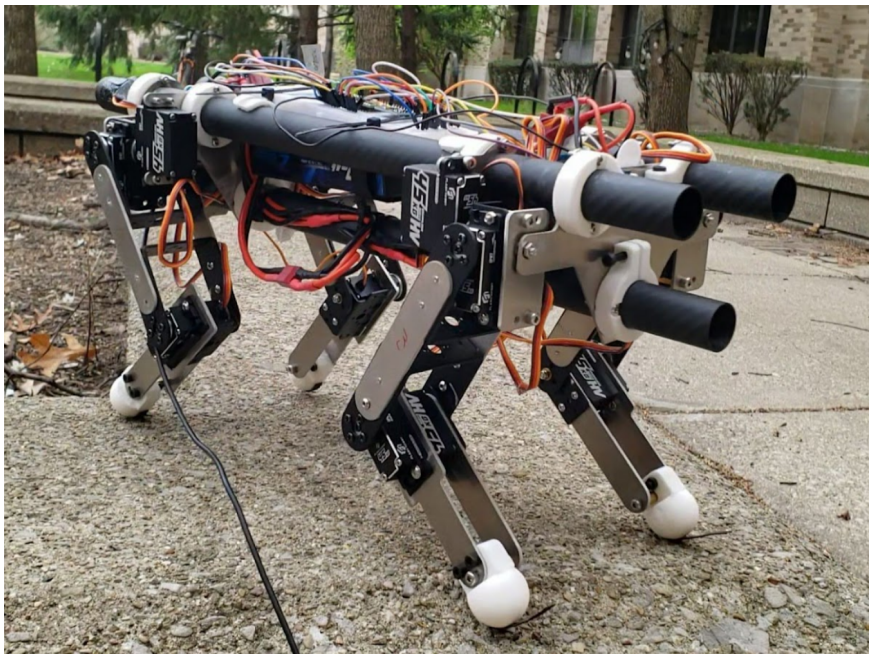

Figure 4: Photo of the Final Robot

It effectively met the goals initially set out for this project. The dog can walk forward & backward and turn left & right without falling over. The balance controller is effective enough that it can even handle some uneven surfaces. It takes serial inputs sent from a laptop to determine what direction to move.

There are still a number of issues with the robot. First, it is not completely remote control. The dog is still tethered to a laptop to send commands. This would make the robot much easier to use. Second, electrical noise occasionally interferes with the signals to the motors, causing them to glitch out. Shielding and shortening the signal wires would make leg movements more consistent and predictable. Third, the balance control is still rudimentary. If a more sophisticated controller was used, the robot could move more smoothly and handle significantly rougher terrain.

Overall, I am pleased with the results for this project.

References:

[1] James Bruton Youtube Channel, https://youtu.be/aDlHz0W0R44
[2] James Bruton Github, https://github.com/XRobots/miniDogV2
[3] MPU6050 Github, https://github.com/electroniccats/mpu6050