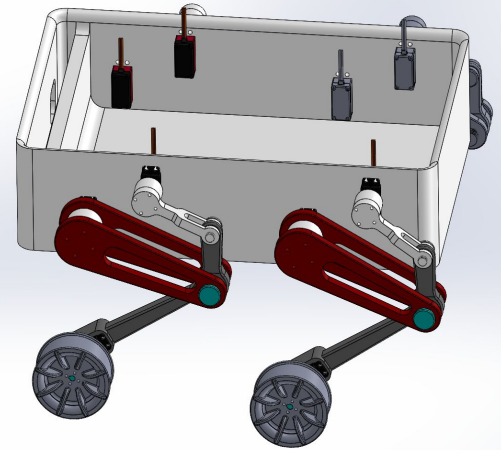


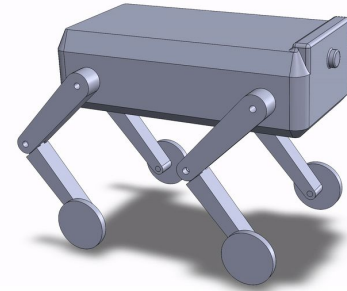
WALT

Wheeled Autonomous Locomotion Traveler

Leon Greiner, Vineeth Parashivamurthy,
Yichen Hu, Yitong Wu, Zichu Zhou



Basic Idea & Objectives



Autonomous Wheeled Quadruped Robot

- **Dual-Mode locomotion:** Walking/Climbing and Driving.
- Each foot is equipped with **powered wheels**, that can be locked in walking mode
- **8-DOF** Leg System with 4 Powered Wheels

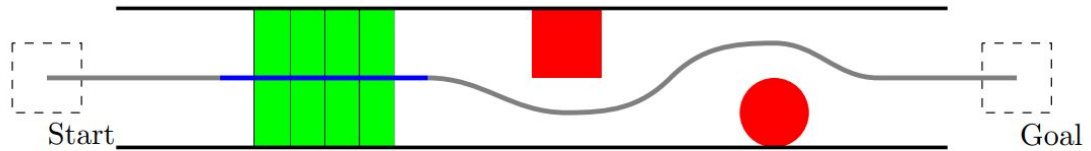
Objective 1: Basic locomotion and mode switching.

Objectives 2: Autonomous stair detection and climbing.

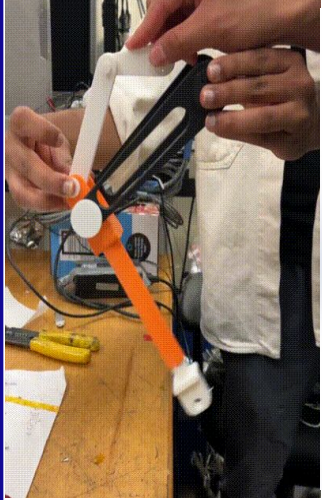
Objectives 3: Autonomous driving and climbing with integrated obstacle avoidance and path planning.

Legend:

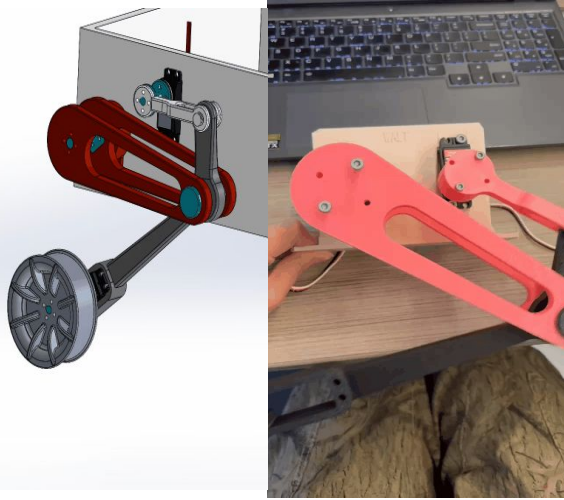
- Passed Obstacles
- Climbed Stairs
- Driving Path
- Walking/Climbing Path



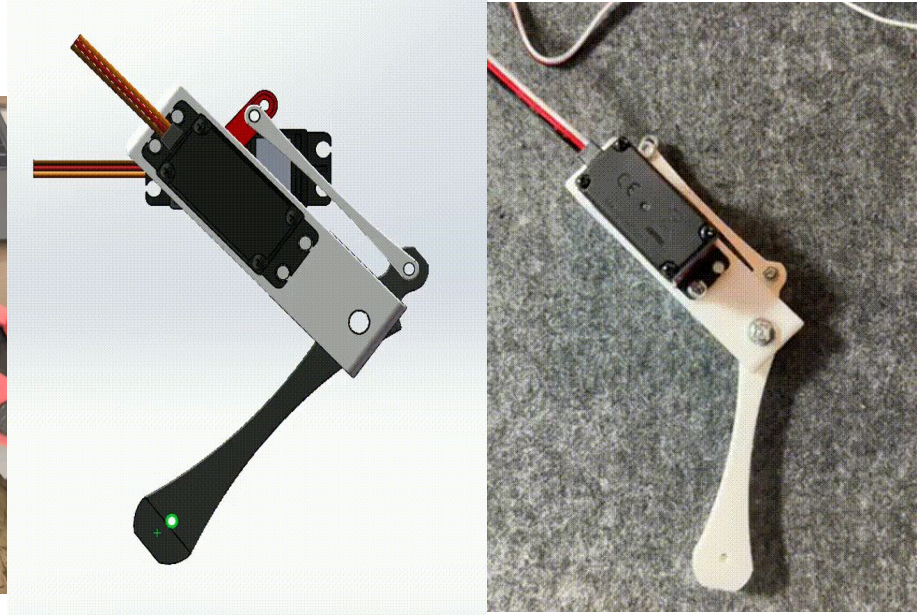
Mechanical Design



Prototype 1



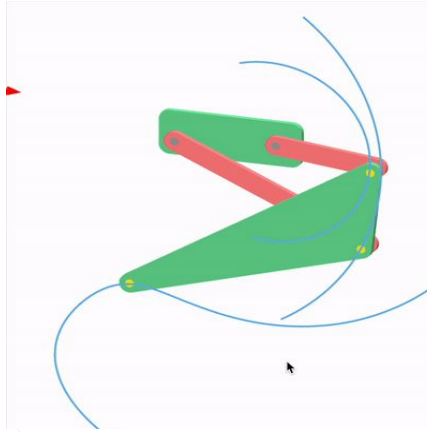
Prototype 2



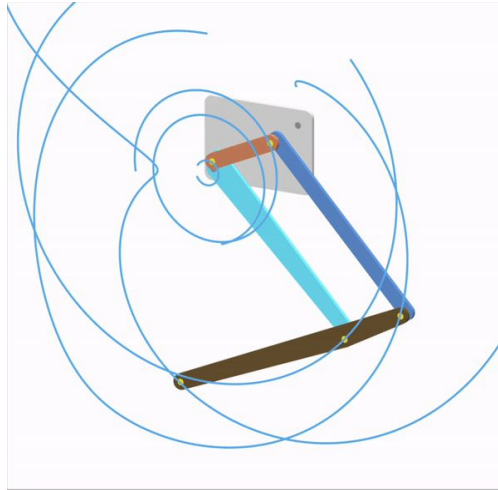
Final Design

Dynamics Model and Simulation

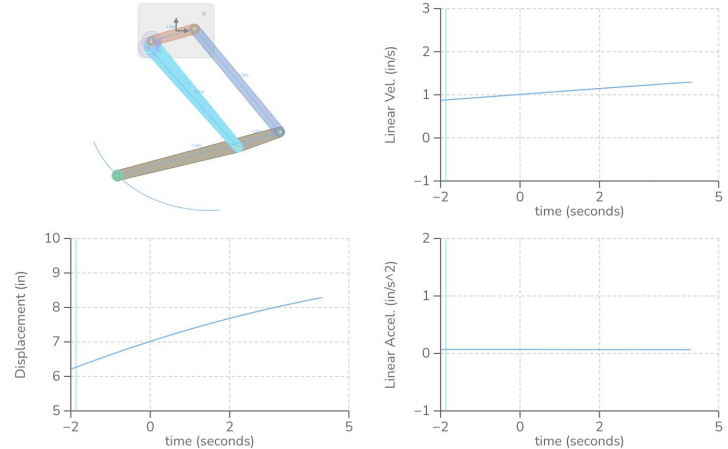
1. We utilized **MotionGen**, a software tool, to synthesize and validate kinematic linkages iteratively to achieve the desired output.
2. The software enables the determination of joint angles and velocities, which can be further employed in Lagrangian dynamics to calculate the torque requirements for each joint efficiently.



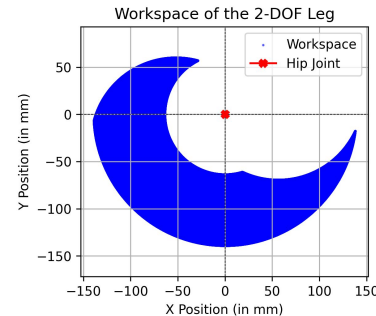
Initial Kinematic prototype



Kinematics of final design



Velocity, and acceleration plots to calculate Torque

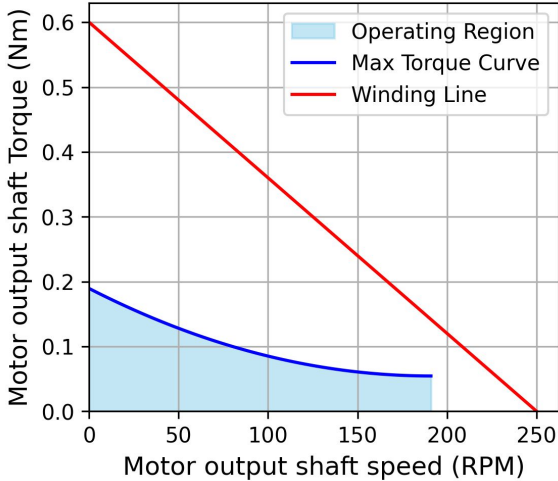


Workspace of Robot

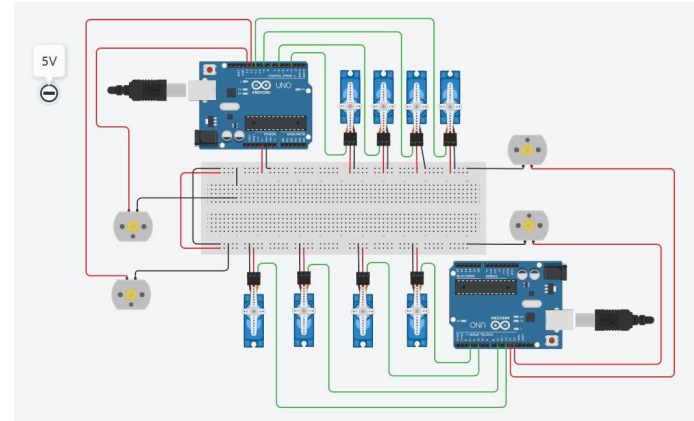
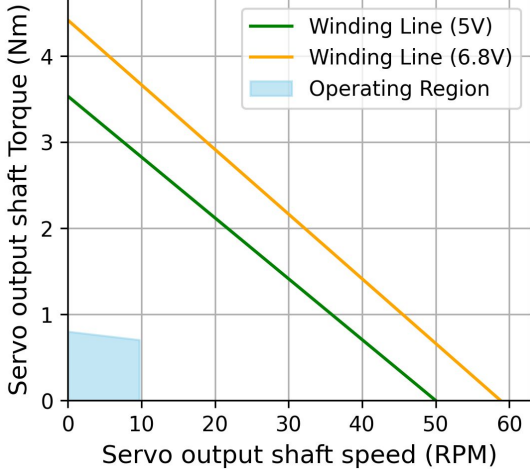
Both servos and performance cu

The graph illustrates the operating region for the motor. The y-axis represents Motor output shaft torque in Nm, ranging from 0.0 to 0.6. The x-axis represents Motor output shaft speed in RPM, ranging from 0 to 250. The operating region is shaded in light blue, bounded by the max torque curve (blue line) and the winding line (red line). The max torque curve starts at approximately 0.19 Nm at 0 RPM and decreases to about 0.05 Nm at 190 RPM. The winding line starts at 0.6 Nm at 0 RPM and decreases linearly to 0 Nm at 250 RPM.

Motor output shaft speed (RPM)	Max Torque Curve (Nm)	Winding Line (Nm)
0	0.19	0.60
50	0.13	0.48
100	0.09	0.36
150	0.06	0.24
190	0.05	0.12
250	-	0.00

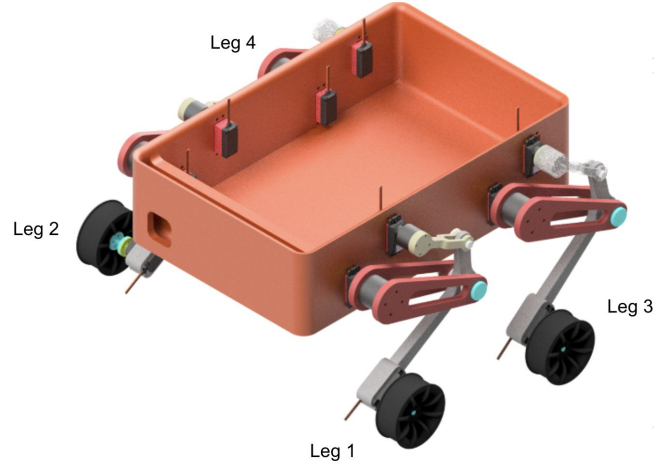


This graph shows the relationship between servo output shaft torque and speed for a 5V servo. The y-axis represents Servo output shaft Torque (Nm) from 0 to 4. The x-axis represents Servo output shaft speed (RPM) from 0 to 60. A green line, labeled 'Winding Line (5V)', shows a linear decrease in torque from approximately 3.5 Nm at 0 RPM to 0 Nm at 50 RPM. An orange line, labeled 'Winding Line (6.8V)', shows a linear decrease from approximately 4.5 Nm at 0 RPM to 0 Nm at 60 RPM. A light blue shaded area, labeled 'Operating Region', is bounded by the y-axis, the 5V winding line, and a horizontal line at approximately 0.8 Nm, extending from 0 RPM to about 10 RPM.



- Extracted joint angles and their derivatives from the simulated gait cycle to calculate the Lagrangian, determining the **torque-speed requirements** for the servos.
- Calculated **wheel motor requirements** based on a desired decreasing acceleration profile relative to speed.
- Both servos and wheel motors **meet the derived requirements**, as confirmed by their performance curves matching the respective winding lines.

Quadruped Hybrid Movement Algorithm



Algorithm 1 QuadrupedHybridMovement

```

1: function QUADRUPEDMOVE(mode)
2:   Let  $L = \{L_1, L_2, L_3, L_4\}$  be legs
3:   Let  $\theta_s \in \mathbb{R}^3$  be servo angles per leg
4:   Let  $v \in \mathbb{R}$  be wheel velocity

   Define sequences  $S(\text{mode})$ :
5:   Forward:

$$S_f = [(\{L_1, L_4\}, \theta^\uparrow, 0.5s) \rightarrow (\{L_1, L_4\}, \theta^\rightarrow, 0.5s) \rightarrow (\{L_1, L_4\}, \theta^\downarrow, 0.5s) \\ \rightarrow (\{L_2, L_3\}, \theta^\uparrow, 0.5s) \rightarrow (\{L_2, L_3\}, \theta^\rightarrow, 0.5s) \rightarrow (\{L_2, L_3\}, \theta^\downarrow, 0.5s)]$$

6:   Wheel:

$$S_w = [(v^\rightarrow, 1.0s)]$$

7:   Stairs:

$$S_s = [(L_1, \theta_h^\uparrow, 0.6s) \rightarrow (L_1, \theta_h^\rightarrow, 0.6s) \rightarrow (L_1, \theta_s^\downarrow, 0.6s) \\ \rightarrow (L_2, \theta_h^\uparrow, 0.6s) \rightarrow (L_2, \theta_h^\rightarrow, 0.6s) \rightarrow (L_2, \theta_s^\downarrow, 0.6s) \\ \rightarrow (v^\nearrow, 1.0s) \\ \rightarrow (L_3, \theta_h^\uparrow, 0.6s) \rightarrow (L_3, \theta_h^\rightarrow, 0.6s) \rightarrow (L_3, \theta_s^\downarrow, 0.6s) \\ \rightarrow (L_4, \theta_h^\uparrow, 0.6s) \rightarrow (L_4, \theta_h^\rightarrow, 0.6s) \rightarrow (L_4, \theta_s^\downarrow, 0.6s)]$$

8:   TurnLeft:

$$S_{tl} = [(v_L = v_{slow}, v_R = v_{fast}, 1.0s)]$$

9:   TurnRight:

$$S_{tr} = [(v_L = v_{fast}, v_R = v_{slow}, 1.0s)]$$


   Execute( $n_{steps}$ ):
10:  for step  $\in [1..n_{steps}]$  do
11:    for all action  $\in S(\text{mode})$  do
12:      if action.type = leg then
13:        for all  $l \in \text{action.legs}$  do
14:           $\theta_s[l] \leftarrow \text{action.angle}$ 
15:          updateServo( $l, \theta_s[l]$ )
16:        end for
17:      else
18:        updateWheels(action.v)
19:      end if
20:      delay(action.t)
21:    end for
22:  end for
23: end function

```

Where:

- θ^\uparrow = lift angle
- θ^\rightarrow = forward swing angle
- θ^\downarrow = lower angle
- θ_h^\uparrow = high lift angle
- θ_h^\rightarrow = high forward swing angle
- θ_s^\downarrow = stair lower angle
- v^\rightarrow = forward velocity
- v^\nearrow = incline velocity

Control

$$e_x = x_{desired} - x_{actual}, \quad e_y = y_{desired} - y_{actual}$$

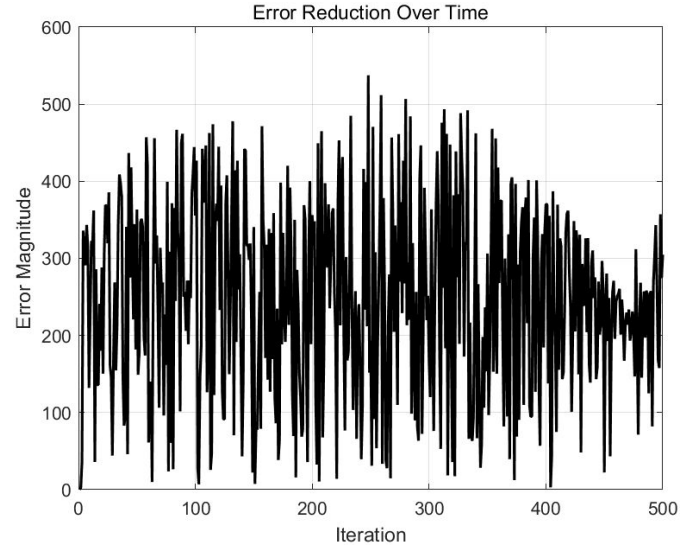
$$E = \sqrt{e_x^2 + e_y^2}$$

To minimize the error E , the control inputs u_x and u_y are defined as:

$$u_x = K_x \cdot e_x + K_c \cdot e_y, \quad u_y = K_y \cdot e_y + K_c \cdot e_x$$

Dynamic gain adjustment is introduced as:

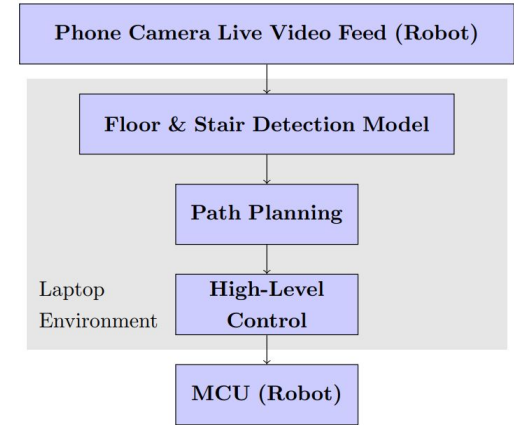
$$K_x = K_x^o \cdot (1 + \alpha \cdot E), \quad K_c = K_y^o \cdot (1 + \alpha \cdot E)$$



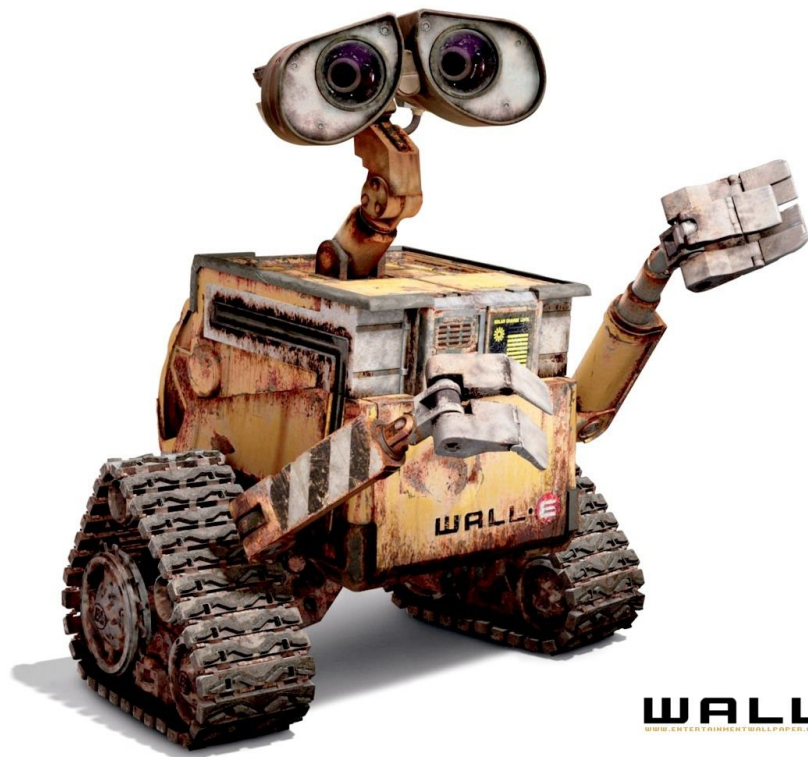
- **Cross Coupling Control(CCC):** The goal is to minimize the error between the desired trajectory (C_desired) and the actual trajectory (C_actual).
- **Dynamic gain adjustment** is introduced to modify the control gains based on the magnitude of the error, allowing faster response when the error is large and greater stability when the error is small.

Perception & Path Planning

- **Video Feed:** Processes RGB camera data via a smartphone mounted on the robot.
- **Detection Model:** YOLOv11 detects drivable surfaces and obstacles, by segment floor and stairs.
- **Path Planning:** Generates an optimal path by following the corridor's centerline, maintaining sufficient distance to walls and obstacles.
- **High-Level Control:** Calculates the turning rate using an enhanced lookahead algorithm combined with a PID controller and stair proximity based on the distance to detected stairs, triggering predefined climbing control.
- **Testing Performance:** Achieves real-time processing at over 10 FPS on a laptop, ensuring smooth and efficient corridor navigation.



Q & A



WALL·E
WWW.DOLBY DIGITAL WALL·E.COM