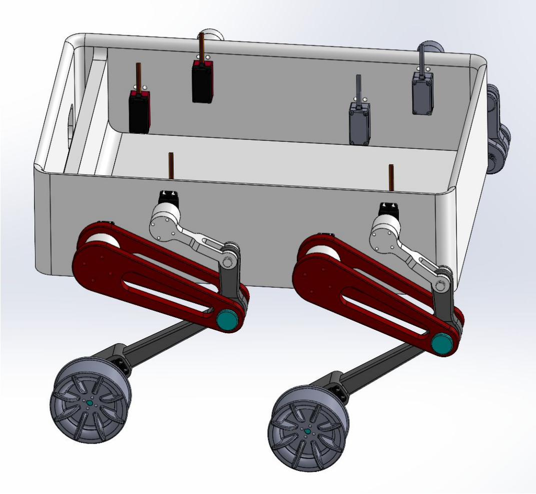
WALT

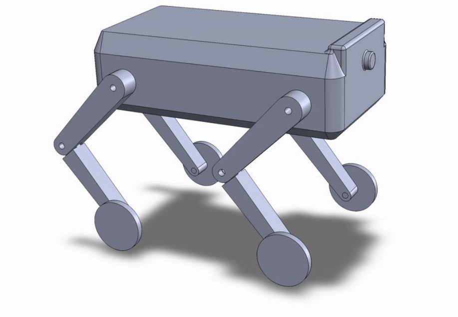
Wheeled Autonomous Locomotion Traveler

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



Group 6 | MECENG 239 | Final Presentation

Basic Idea & Objectives



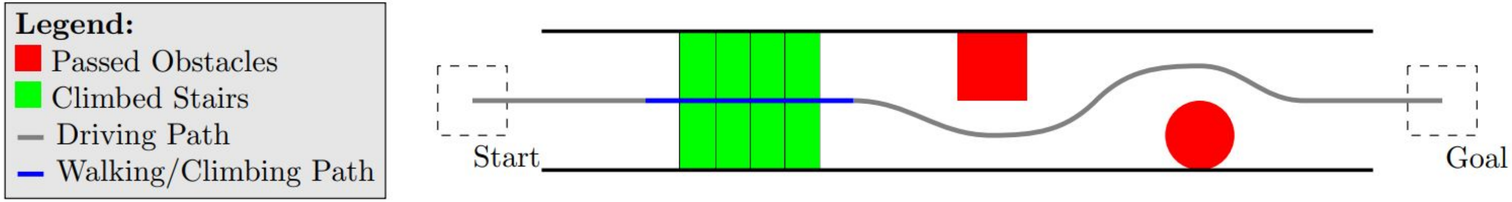
Autonomous Wheeled Quadruped Robot

- **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

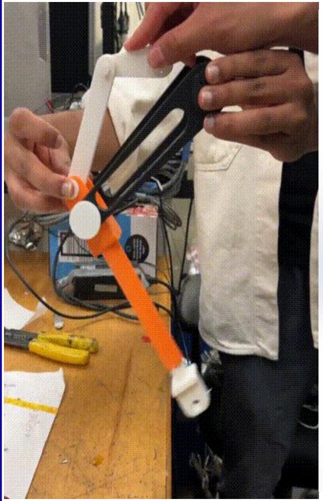
Objective 1: Basic locomotion and mode switching.

Objectives 2: Autonomous stair detection and climbing.

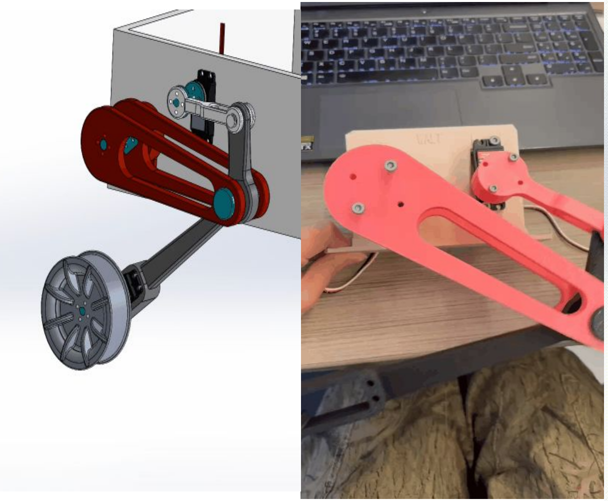
Objectives 3: Autonomous driving and climbing with integrated obstacle avoidance and path planning.



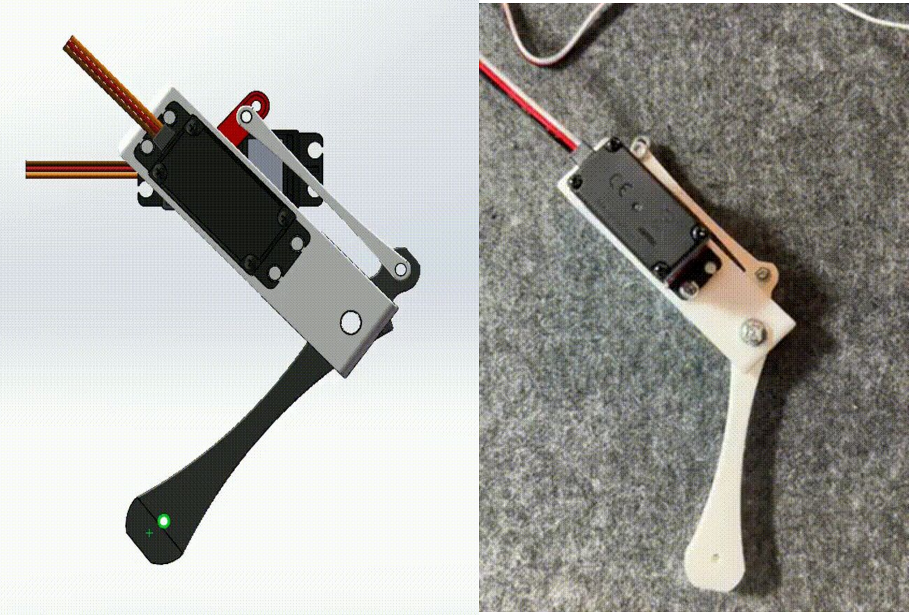
Mechanical Design



Prototype 1



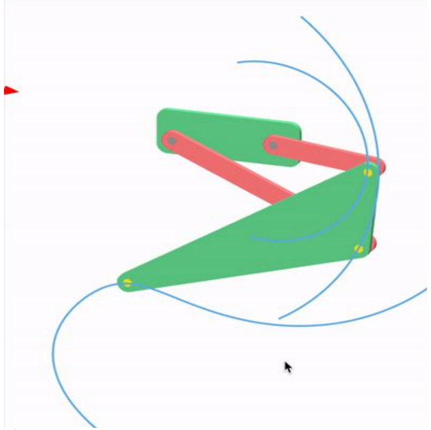
Prototype 2



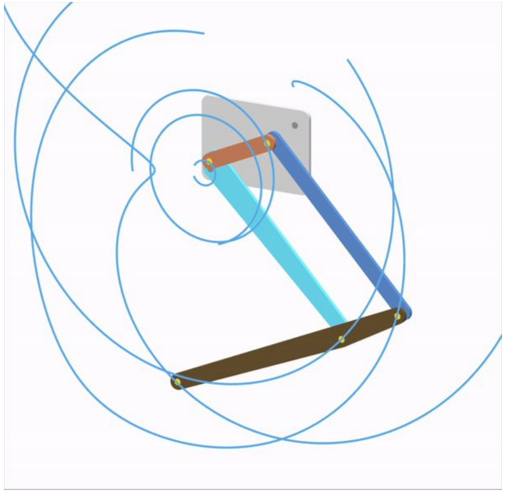
Final Design

Dynamics Model and Simulation

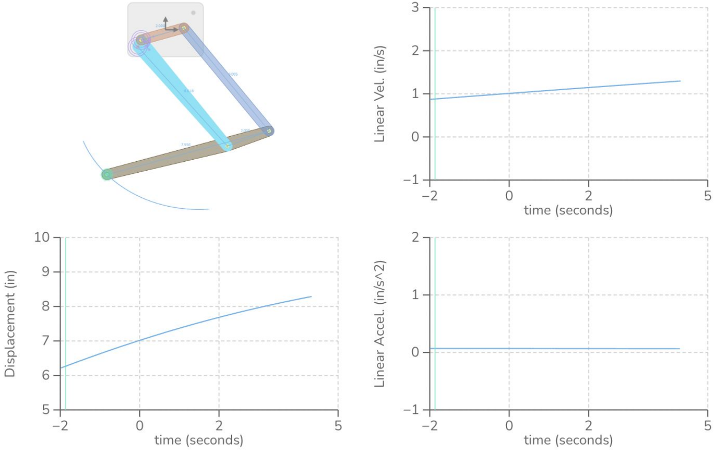
1. We utilized **MotionGen**, a software tool, to synthesize and validate kinematic linkages iteratively to achieve the desired output.
2. The software enables the determination of joint angles and velocities, which can be further employed in Lagrangian dynamics to calculate the torque requirements for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



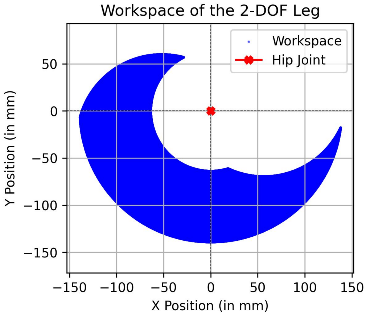
Initial Kinematic prototype



Kinematics of final design

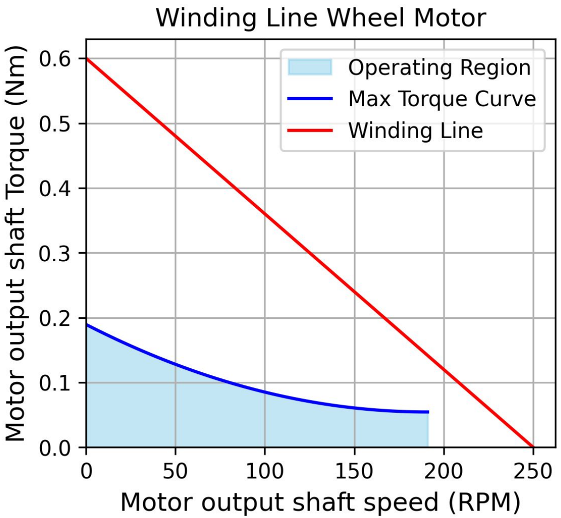


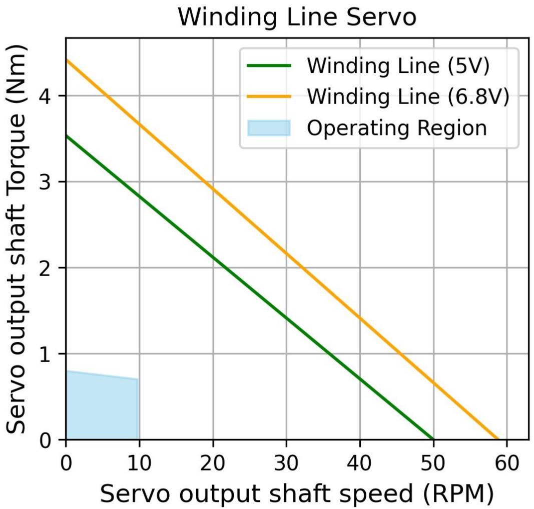
Velocity, and acceleration plots to calculate Torque

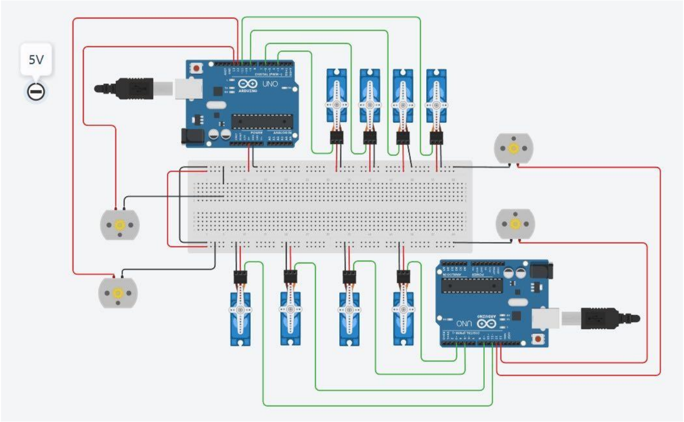


Workspace of Robot

Electronics

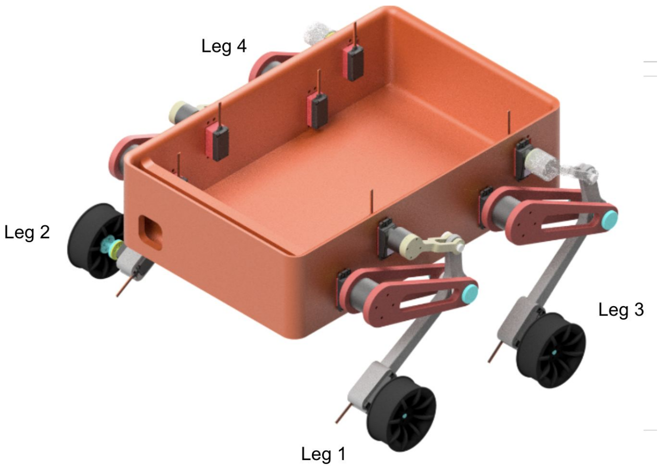






- Extracted joint angles and their derivatives from the simulated gait cycle to calculate the Lagrangian, determining the **torque-speed requirements** for the servos.
- Calculated **wheel motor requirements** based on a desired decreasing acceleration profile relative to speed.
- Both servos and wheel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

Quadruped Hybrid Movement Algorithm



Algorithm 1 QuadrupedHybridMovement

function QUADRUPEDMOVE(mode)
  Let $L = \{L_1, L_2, L_3, L_4\}$ be legs
  Let $\boldsymbol{\theta}_s \in \mathbb{R}^3$ be servo angles per leg
  Let $v \in \mathbb{R}$ be wheel velocity

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

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

  Wheel:

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

  Stairs:

$$S_s = [(L_1, \boldsymbol{\theta}_h^{\uparrow}, 0.6s) \to (L_1, \boldsymbol{\theta}_h^{\rightarrow}, 0.6s) \to (L_1, \boldsymbol{\theta}_s^{\downarrow}, 0.6s) \to (L_2, \boldsymbol{\theta}_h^{\uparrow}, 0.6s) \to (L_2, \boldsymbol{\theta}_h^{\rightarrow}, 0.6s) \to (L_2, \boldsymbol{\theta}_s^{\downarrow}, 0.6s) \to (v^{\nearrow}, 1.0s) \to (L_3, \boldsymbol{\theta}_h^{\uparrow}, 0.6s) \to (L_3, \boldsymbol{\theta}_h^{\rightarrow}, 0.6s) \to (L_3, \boldsymbol{\theta}_s^{\downarrow}, 0.6s) \to (L_4, \boldsymbol{\theta}_h^{\uparrow}, 0.6s) \to (L_4, \boldsymbol{\theta}_h^{\rightarrow}, 0.6s) \to (L_4, \boldsymbol{\theta}_s^{\downarrow}, 0.6s)]$$

  TurnLeft:

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

  TurnRight:

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

  Execute(n_{steps}):
  for step $\in [1..n_{steps}]$ **do**
    for all action $\in S(\text{mode})$ **do**
      if action.type = leg **then**
        for all $l \in$ action.legs **do**
          $\boldsymbol{\theta}_s[l] \leftarrow$ action.angle
          updateServo(l, $\boldsymbol{\theta}_s[l]$)
        end for
      else
        updateWheels(action.v)
      end if
      delay(action.t)
    end for
  end for
end function

Where:

- $\boldsymbol{\theta}^{\uparrow}$ = lift angle
- $\boldsymbol{\theta}^{\rightarrow}$ = forward swing angle
- $\boldsymbol{\theta}^{\downarrow}$ = lower angle
- $\boldsymbol{\theta}_h^{\uparrow}$ = high lift angle
- $\boldsymbol{\theta}_h^{\rightarrow}$ = high forward swing angle
- $\boldsymbol{\theta}_s^{\downarrow}$ = stair lower angle
- $v^{\rightarrow}$ = forward velocity
- $v^{\nearrow}$ = incline velocity

Control

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

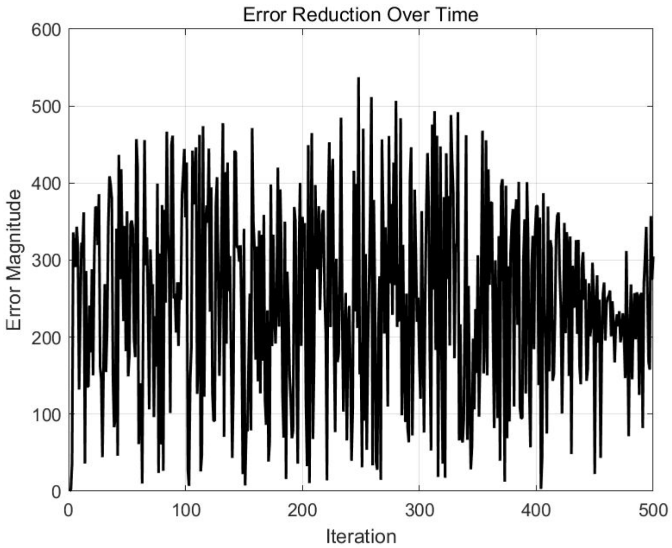
$$E = \sqrt{e_x^2 + e_y^2}$$

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:

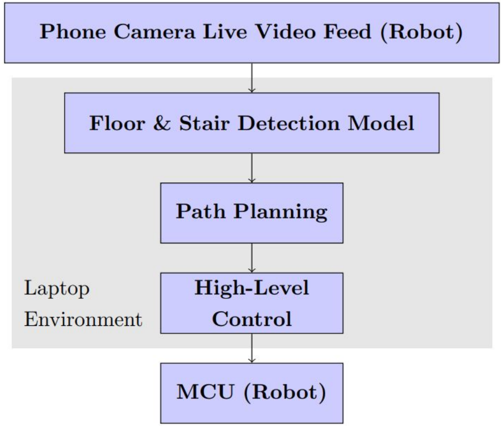
$$K_x = K_x^o \cdot (1 + \alpha \cdot E), \quad K_c = K_y^o \cdot (1 + \alpha \cdot E)$$



- **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