ECE 445

SENIOR DESIGN LABORATORY

DESIGN DOCUMENT

Final Report for ECE445

Team #25

JIAJUN HU (jiajunh5)
YIXUAN LI (yixuan19)
YUHAO WANG (yuhaow7)
XUCHEN DING (xuchend2)

TA: Leixin Chang

March 22, 2024

Contents

1 Introduction

1.1 Problem

Modern air travel involves the constant movement of passengers within airport premises, often burdened with personal belongings and luggage. While airports strive to provide convenience, the process of moving from the check-in area to the departure gate can be challenging for passengers, especially those carrying heavier bags or many bags. This challenge is particularly relevant given the increasing trend of passengers bringing additional luggage, with weights ranging from 2 to 3 kilograms. Although the total weight might still in under the restriction of 50 lbs, the increased amount of bags will add difficulties for people to hold with two hands.

1.2 Solution

The proposed solution for this problem is the development of a leg-wheeled robotic system designed to accompany airport passengers with their luggage. The primary objective is to enhance the passenger experience by offering a reliable and autonomous companion capable of carrying bags weighing 2-3 kilograms. This robotic system will intelligently follow passengers with the help of camera and vision control algorithms as they traverse the airport, providing a hands-free and effortless solution to the burden of carrying personal items. In instances where passengers face challenges, such as staircases, the robot's unique legged design allows it to overcome obstacles that traditional wheeled robots cannot. This robustness ensures smooth navigation in a variety of airport environments.

1.3 Visual Aid

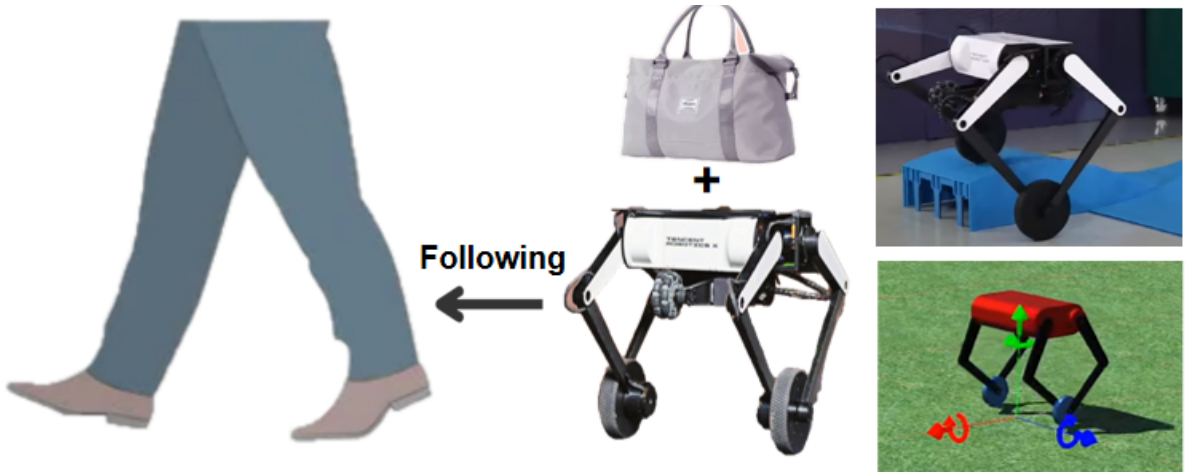


Figure 1: Overall Work Flow

1.4 High-level requirements list

1. The luggage weight must be around 2-3 kg.

2. The overall size of the robot needs to be around 500mm×500mm.
3. The distance between the legs and the robot should be within 1m throughout the whole process.

As shown below, our design contains the following part: Power unit, the control unit, the planning unit, sensor unit and motor unit. The control unit is the central unit of our system, where the microcontroller receives the command and execute by sending signals to different motors. The power unit is responsible for converting the battery voltage from 24v to any voltage needed by different units. The motoe unit consists of four leg motors and two wheel motors.

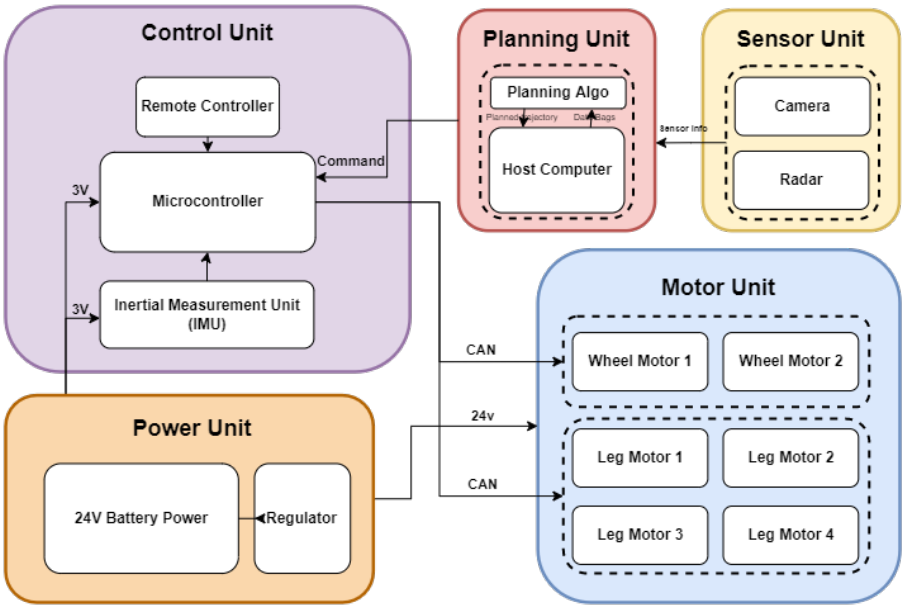


Figure 2: Top Level

2 Design

2.1 Design Procedure

2.1.1 Mechanical Structure

First we determined the overall size of the robot needs to be around 500mm×500mm, so it can carry a normal-sized handbag. Second, we select proper leg types. We choose a 5-bar linkage leg type, because this structure has been well researched and there are plenty of open source about its control methods. Third for stability, we focus on the leg joints. Many wheeled leg robots exhibit a splaying of the feet, which is due to gaps in their leg joints, causing the robot's lower legs to bend outward under high torque. Our solution is to increase the diameter of the cylindrical shaft of the leg joint to suppress the splaying phenomenon.

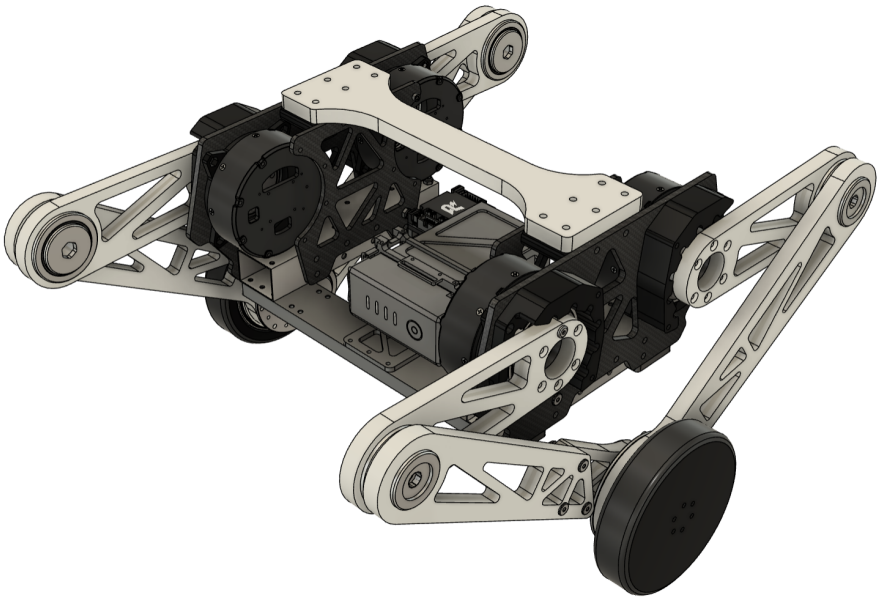


Figure 3: Robot CAD in Fusion

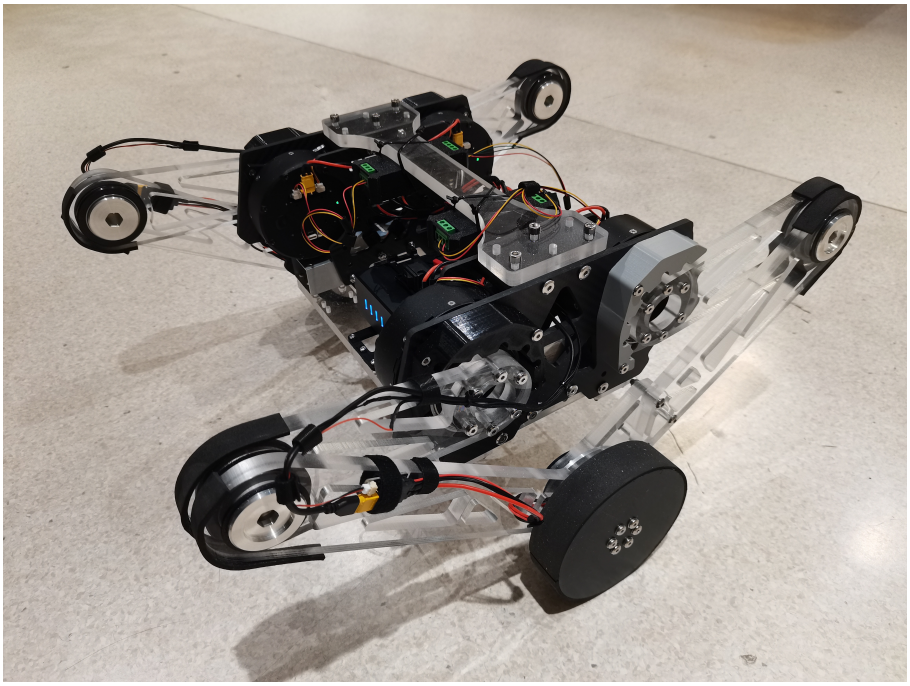


Figure 4: Robot model in reality

2.1.2 Mechanical System

Motor selection

First, we need to select the right joint motors. We assume the weight of our balance robot with carried weight is totally of 20kg, and then calculate the approximate torque at hip joint is around 8Nm. Therefore, we use Unitree Technology's A1 joint motor, which has a peak torque of 33.5 Nm. In CAD model, we assigned the corresponding

material to each part and got the predicted weight is 8.35kg. Assigning material properties also helps us directly get the position of the center of mass. The real weight we measured is 8.2kg which matches our CAD surprisingly well.

Joint Design

At the Knee joints, we use CNC machined 6061-T6 aluminum parts and bowl group bearings for joint fixation. This method allows for a larger joint shaft diameter, effectively reducing the backlash in the leg joints. It also minimizes the parts number we need.

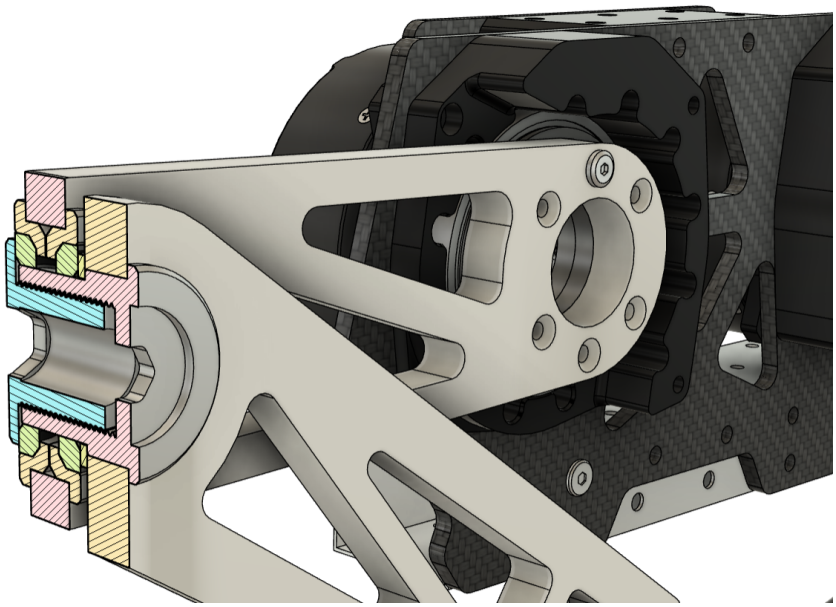


Figure 5: Knee joint cross-section view

At the wheel joints, we use XIAOMI's Cyber Gear motors. Two 3D-printed parts are introduced to lock the position of the KP035 bearing. The wheel is connected to the wheel motor, and the step-shape structure inside the wheel will contact with the bearing. This helps release the load on the motors' rotor.

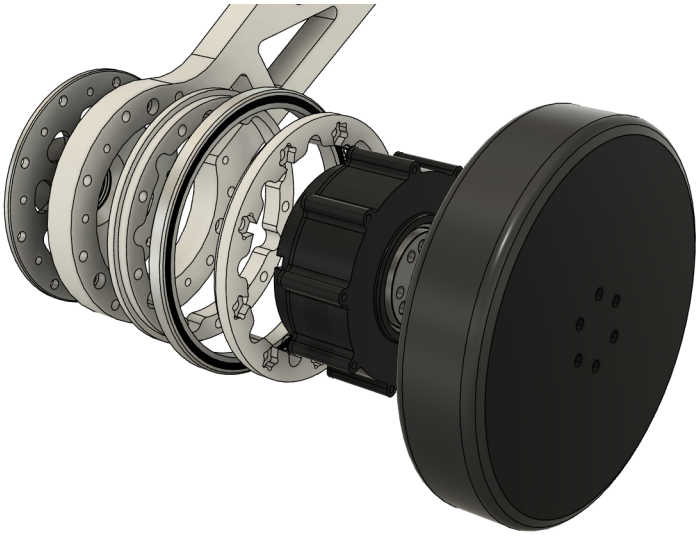


Figure 6: Wheel joint explosion view

At the hip joint, although the Unitree motor is able to bear huge loads, we still decided to add a bearing to release the force act on the motor. Besides, we have to design a physical restriction for the hip joint. This is because Unitree motor has only a single encoder. Therefore, after going through a gear ratio of 9 speed reduction, the outside rotor would have 9 zero points. So we plan to retract the leg when powering the robot. Once the leg hits the upper restriction, the robot will record this as its initial position.

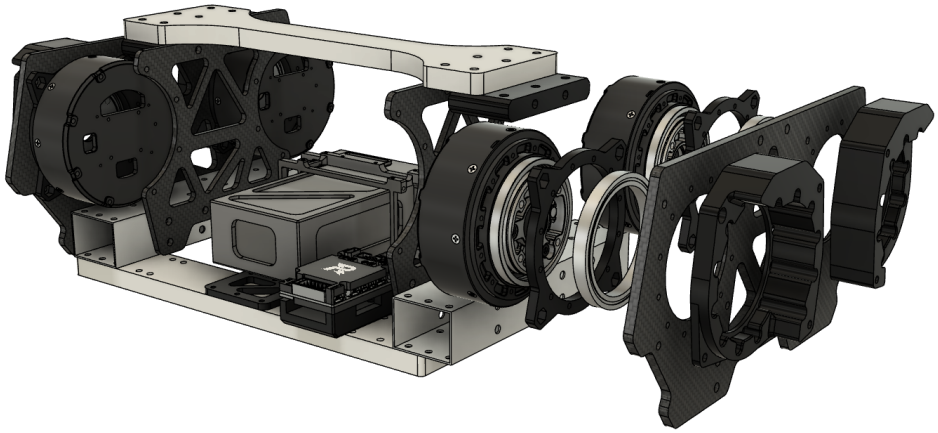


Figure 7: Hip joint explosion view

Material

In the iteration process, we use cheap 3D-printed PLA parts and acrylic boards to build prototypes. Even the knee joints and hip-bearing seats are printed PLA for tolerance

testing. To fix joint motors, we use two boards to sandwich joint motors and attach them to an aluminum square tube. The aluminum square tube functions as corner pieces, which is convenient for connecting boards with 90° angle.

Our final robot uses carbon fiber materials for body structure because they have a lighter mass compared to other materials at the same strength. We also planned to use carbon fiber legs at the beginning. However, due to budget problems, we eventually kept the 12mm-thick acrylic legs. In the progress of building this robot, we found the acrylic boards aren't as fragile as we used to believe. We also adopted topology analysis to reduce as much weight as possible.

We are also using topology to minimize our robot's weight and maintain the parts' strength at the same time. The following is the topology optimization result in Fusion 360. By using this techniques, we are able to reduce around 30 percent of the robot's weight.

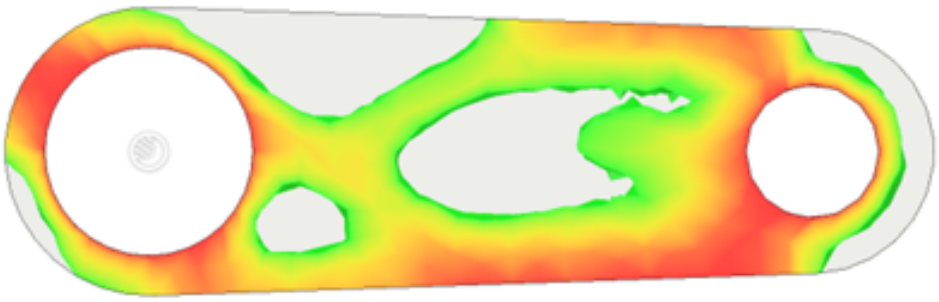


Figure 8: Upper leg topology result

2.1.3 Electrical System

Micro-controller Board

We are using STM32F407IGH6 for our microcontroller board. This is a new product developed by ST in 2011 following the ARM Cortex-M4 architecture. Compared with the previous products, the STM32F4's new integrated FPU unit and DSP instructions greatly enrich the functions of the STM32 chip, and at the same time, the STM32F4's main frequency has also been improved. The main frequency can reach up to 168Mhz (can have 210DMIPS running speed), making the STM32F4 in floating point computing or DSP processing ability greatly improved, with a very wide range of application prospects.

Motor Controll

We are using the joint A1 motor from Unitree and the wheel motor from XIAOMI. The A1 motor uses RS485 to communicate and the wheel motor uses CAN (Controller Area Network) to communicate. Because our micro-controller board doesn't have RS485 port, so we choose to use UART and a TTL to RS485 converter to control A1 motor.

Embedded Software

To communicate with the peripherals, we use the standard API and the API from HAL. What's more, we decided to use a Real-time operating system to support the multi-threads process. Therefore, we decided to use STM32CubeMX to generate the basic code skeleton, and use VS Code as our code editor for its powerful add-ons. We choose GCC, GDB, and OpenOCD as our toolchain and debug tool.

2.1.4 Control System

For the control system, the strategy mainly consists of two parts: the wheel part and the leg part. As suggested in [1], we use the Linear quadratic regulator (LQR) algorithm for the wheel part and use virtual model contorl algorithm (VMC) for the leg part. We choose LQR because the mathematical principles of LQR are relatively simple to implement and it owns rapid response to system changes and strong robustness. And the reason for using VMC is that the VMC algorithm comprehensively considers both the posture and velocity of the robot without the need for inverse kinematics calculations. It achieves high computational and control efficiency. Our design was tested on matlab simulink and Webot for simulation and later on moved to the real robot.

2.2 Design Details

2.2.1 Control System of Wheels

The forward motion of the wheel can be decomposed into forward motion and relative rotation (pitch) around the center of mass P of the car body.
The force analysis diagram for the left and right wheels of the two-wheeled robot is shown in the figure 9. The resultant force in the horizontal direction for the robot is the vector sum of the frictional force F_r between the wheels and the ground, as well as the horizontal force H_r acting between the chassis and the wheels. Since parameters (mass, moment of inertia, radius) being the same for both wheels. Let's conduct force analysis using the example of the right wheel:

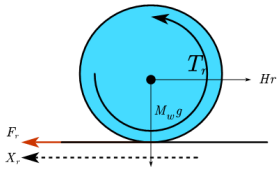


Figure 9: Wheel model

Horizontally, we can derive the equation that:

$$\ddot{x}_r = \frac{F_r - H_r}{M_w} \tag{1}$$

$$\dot{\omega_r} = \frac{T_r - F_r R}{I} \tag{2}$$

Suppose $\omega_r = \frac{\dot{x_r}}{r}$, where r is the radius of each wheel, and combine (1) and (2). We can obtain that :

$$(M_w + \frac{I}{r^2})\ddot{x_r} = \frac{T_r}{r} - H_r \tag{3}$$

The horizontal displacement of the center O of the car chassis x is:

$$x = \frac{x_l + x_r}{2} \tag{4}$$

Therefore, applying equation (3) to the left wheel and combine them together, we could obtain the equation 5:

$$(M_w + \frac{I}{r^2})\ddot{x} = \frac{T_l + T_r}{2r} - \frac{H_l + H_r}{2} \tag{5}$$

Moving forward and backward

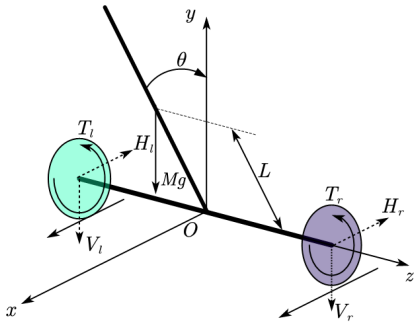


Figure 10: Moving forward and backward

For the vehicle body, according to Newton's second law, we can derive the equation horizontally:

$$H_l + H_r = M\frac{d(x + Lsin\theta)}{dt^2} \tag{6}$$

and vertically:

$$V_l + V_r = M\frac{d(Lcos\theta)}{dt^2} + Mg \tag{7}$$

For the vehicle body, according to the rigid body rotation law with a fixed axis, we can obtain the equation:

(8)

Combining (6), (7) and (2.2.1), we could obtain:

$$\ddot{\theta} = \frac{gsin\theta - \ddot{x}cos\theta - (\frac{T_l+T_r}{ML})}{\frac{J}{ML} + L} \tag{9}$$

Combining the equation (6) and equation (5), we can get:

$$(M + 2M_w + \frac{2I}{r^2})\ddot{x} = ML(\ddot{\theta}cos\theta - \dot{\theta}^2 sin\theta) + \frac{T_l + T_r}{2} \tag{10}$$

When our vehicle body can keep balance and steady with a neglectable pitch angle θ, we have the approximation as following:

$$cos\theta = 1, sin\theta = \theta, \dot{\theta^2} = 0 \tag{11}$$

Then the equation (10) can be linearized as :

$$\ddot{x} = \frac{T_l + T_r}{(M + 2M_w + \frac{2I}{r^2})r} - \frac{ML\ddot{\theta}}{(M + 2M_w + \frac{2I}{r^2})} \tag{12}$$

And the equation (9) could be linearized as:

$$\ddot{\theta} = \frac{g\theta - \ddot{x} - (\frac{T_l+T_r}{ML})}{\frac{J}{ML} + L} \tag{13}$$

Solving the equation system composed of (12) and (13), we get the following formula at last:

$$\ddot{x} = \frac{J + ML^2 + MLr}{JMr + r(J + ML^2)(2M_w + \frac{2I}{r^2})}(T_l + T_r) - \theta\frac{M^2L^2g}{JM + (J + ML^2)(2M_w + \frac{2I}{r^2})} \tag{14}$$

$$\ddot{\theta} = \theta\frac{MLg(M + 2M_w + \frac{2I}{r^2}}{JM + (J + ML^2)(2M_w + \frac{2I}{r^2})} - \frac{\frac{ML}{r} + M + 2M_w + \frac{2I}{r^2}}{JM + (J + ML^2)(2M_w + \frac{2I}{r^2})}(T_l + T_r) \tag{15}$$

Rotation Motion

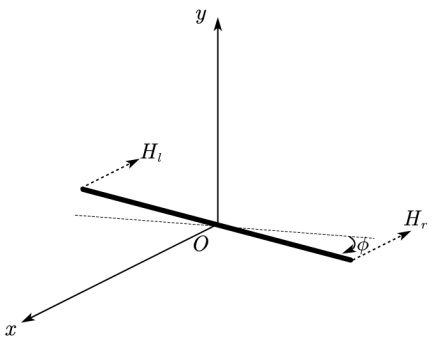


Figure 11: Rotation model

According to the law of rigid body rotation, we have :

$$J_\phi\ddot{\phi} = \frac{D}{2}(H_l - H_r) \tag{16}$$

$$\ddot{\phi} = \frac{\ddot{x}_l - \ddot{x}_r}{2} \tag{17}$$

Then we can plug (3) into the equations above, resulting in:

$$\ddot{\phi} = \frac{T_l - T_r}{rM_wD + \frac{ID}{r} + \frac{2rJ_\phi}{D}} \tag{18}$$

LQR control system

With (12), (13) and (18, we can generate the state function of our control system in the form of $\dot{X} = AX + Bu$:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & A_{23} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & A_{43} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ B_{21} & B_{22} \\ 0 & 0 \\ B_{41} & B_{42} \\ 0 & 0 \\ B_{61} & B_{62} \end{bmatrix} * \begin{bmatrix} T_l \\ T_r \end{bmatrix} \tag{19}$$

Where,

$$A_{23} = -\frac{M^2L^2g}{JM + (J + ML^2)(2M_w + \frac{2I}{r^2})} \tag{20}$$

$$A_{43} = \frac{MLg(M + 2M_w + \frac{2I}{r^2}}{JM + (J + ML^2)(2M_w + \frac{2I}{r^2})} \tag{21}$$

$$B_{21} = B_{22} = \frac{J + ML^2 + MLr}{r(JM + (J + ML^2)(2M_w + \frac{2I}{r^2}))} \tag{22}$$

$$B_{41} = B_{42} = -\frac{\frac{ML}{r} + M + 2M_w + \frac{2I}{r^2}}{JM + (J + ML^2)(2M_w + \frac{2I}{r^2})} \tag{23}$$

$$B_{61} = -B_{62} = \frac{1}{r(DM_w + \frac{ID}{r^2} + \frac{2J_\phi}{d})} \tag{24}$$

According to LQR, we need to set $u = -KX$, where K denotes the feedback gain matrix and minimize the cost function of LQR control algorithm:

$$J = \frac{1}{2}\int_0^\infty (X^TQX + u^TRu)dt \tag{25}$$

Where Q is the state vector weighting matrix of a semi-positive definite symmetric constant, and R is the control rate weighting matrix of a positive definite symmetric constant. Increasing the weighting matrix Q will decrease the overshoot and settling time of the dynamic process, but it will correspondingly increase the energy consumption of the control inputs. When the coefficients of the R matrix increase, it can reduce the number of input variables of the system, but it will also decrease the response speed of the system. To reduce the energy consumption of the system's control, one can appropriately increase the value of the weighting matrix R. However, if the value

of the weighting matrix R is too large, it will result in too small control energy, which is also disadvantageous for controlling the system.
Our final choose of Q, R and the resulting K is as below:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, R = \begin{bmatrix} 1 & 0 \\ 0 & 0.25 \end{bmatrix},$$

$$K = \begin{bmatrix} 178.280859 & -21.821328 & 102.754738 & 83.116185 & 146.698730 & 6.788157 \\ -64.954540 & -26.935612 & 206.678773 & 104.540048 & -127.912524 & -6.057289 \end{bmatrix}$$

2.2.2 Control System of Legs

Five-Linkages leg model

The support structure of our robot consists of symmetric left and right legs. Fig:12 illustrates a five-linkages model of one leg as an example.

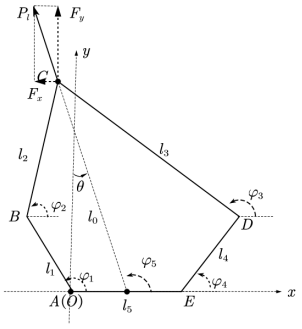


Figure 12: Five-linkage leg model

According to the leg structure, the rotational joints A and E are driven by motors, and their angles can be measured using motor encoders. In our control task, the focus is on the position of the end-point C of the 5-bars linkage mechanism, which can typically be represented using Cartesian (Xc, Yc). The target of the leg control system is to set the wheel point C at the expected position and keep the leg posture normal at the same time. We set point A as the origin of the coordinate system. By solving the five-linkages model for the Cartesian coordinates of point C, we can obtain the following two sets of equations:

$$\begin{cases} x_B + l_2 cos\varphi_2 = x_D + l_3 cos\varphi_3 \\ y_B + l_2 sin\varphi_2 = y_D + l_3 sin\varphi_3 \end{cases} \tag{26}$$

$$\begin{cases} x_B = l_1 cos\varphi_1 \\ y_B = l_1 sin\varphi_1 \\ x_c = l_1 cos\varphi_1 + l_2 cos\varphi_2 \\ y_c = l_1 sin\varphi_1 + l_2 sin\varphi_2 \\ x_D = l_5 + l_4 cos\varphi_4 \\ y_D = l_4 sin\varphi_4 \end{cases} \tag{27}$$

VMC control System

As suggested in [2], we utilize the concept of VMC (Virtual Model Control) in our control system. VMC is a control method that uses virtual forces and components to simulate actual actuator forces and torques. As depicted in the Figure 2.2.2-2, a virtual leg is conceptualized extending to point C. This virtual leg aids in the calculation of the necessary torques at the joints of the physical robot by applying virtual force P_l at point C, which decomposed in the x and y direction i.e. F_x and F_y . φ_5 denotes the angle between l_0 and the chassis AE. By establishing a mathematical relationship between these virtual forces and the real joint torques—specifically at joints A, E—we can derive the torque requirements that ensure the robot's legs move as desired. Next, we define the status of the end-effector (point C) as $x = [x_C, y_C]^T$ and the target control status of Motors A, E as $S = [\varphi_1, \varphi_4]^T$. Thus, we could obtain a forward kinematics fomula for our leg model as following::

$$\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{S}) \tag{28}$$

Take the derivative of this equation, we get:

$$\nabla \boldsymbol{x} = \boldsymbol{J} \nabla \boldsymbol{S} \tag{29}$$

According to the principle of virtual work, the external torque on a system in equilibrium is equal to zero. Therefore we can obtain:

$$\boldsymbol{T}^T \nabla \boldsymbol{S} + (-\boldsymbol{F}^T) \nabla \boldsymbol{x} = 0 \tag{30}$$

Where $\boldsymbol{T} = [T_A, T_E]^T, \boldsymbol{F} = [F_x, F_y]^T$. T_A, T_E are the torques of joint motor A and E respectively. F_x, F_y is the decomposed forces of virtual force P_l at point C along the direction of x and y. Combining (29) and (30), we can have the following formula to transfer our virtual force to the expected actual torque for joints A, E:

$$\boldsymbol{T} = \boldsymbol{J}^T \boldsymbol{F} \tag{31}$$

Take the derivative of x_c, y_c in (27), we can obtain:

$$[\dot{x_c}, \dot{y_c}]^T = [-l_1 \dot{\varphi_1} sin\varphi_1 - l_2 \dot{\varphi_2} sin\varphi_2, l_1 \dot{\varphi_1} cos\varphi_1 + l_2 \dot{\varphi_2} cos\varphi_2]^T \tag{32}$$

Then we take the derivative of (26):

$$\begin{cases} l_3 \dot{\varphi_3} sin\varphi_3 = \dot{x_D} - \dot{x_B} + l_2 \dot{\varphi_2} sin\varphi_2 \\ l_3 \dot{\varphi_3} cos\varphi_3 = \dot{y_B} - \dot{y_D} + l_2 \dot{\varphi_2} cos\varphi_2 \end{cases} \tag{33}$$

Take $\dot{\varphi_3}$ out of (33), we can get the following formula about $\dot{\varphi_2}$:

$$\dot{\varphi_2} = \frac{(\dot{x_D} - \dot{x_B})cos\varphi_3 + (\dot{y_D} - \dot{y_B})sin\varphi_3}{l_2 sin(\varphi_3 - \varphi_2)} \tag{34}$$

Plug (34) into (32):

$$\begin{cases} \dot{x_c} = -l_1\dot{\varphi_1}sin\varphi_1 - l_2 sin\varphi_2 \frac{(\dot{x_D} - \dot{x_B})cos\varphi_3 + (\dot{y_D} - \dot{y_B})sin\varphi_3}{l_2 sin(\varphi_3 - \varphi_2)} \\ \dot{y_c} = l_1\dot{\varphi_1}cos\varphi_1 + l_2 cos\varphi_2 \frac{(\dot{x_D} - \dot{x_B})cos\varphi_3 + (\dot{y_D} - \dot{y_B})sin\varphi_3}{l_2 sin(\varphi_3 - \varphi_2)} \end{cases} \tag{35}$$

Since we can derive the expression of $\dot{x_D}, \dot{x_B}, \dot{y_D}, \dot{y_B}$ with only constant value(l_i) and variable φ_1, φ_4 from (27), with the sum and difference formulas in trigonometry, we get the final equation as following:

$$\begin{bmatrix} \dot{x_C} \\ \dot{y_C} \end{bmatrix} = \begin{bmatrix} sin\varphi_3 \frac{l_1 sin(\varphi_{12})}{sin(\varphi_{23})} & sin\varphi_2 \frac{l_4 sin(\varphi_{34})}{sin(\varphi_{23})} \\ -cos\varphi_3 \frac{l_1 sin(\varphi_{12})}{sin(\varphi_{23}} & -cos\varphi_2 \frac{l_4 sin(\varphi_{34})}{sin(\varphi_{23}} \end{bmatrix} * \begin{bmatrix} \dot{\varphi_1} \\ \dot{\varphi_4} \end{bmatrix} \tag{36}$$

where, $\varphi_{12} = \varphi_1 - \varphi_2, \varphi_{23} = \varphi_2 - \varphi_3, \varphi_{34} = \varphi_3 - \varphi_4$.
According to (30), we derived the following relationship between the expected torque of the joint motor and the virtual force:

$$T_A = l_1 sin\varphi_3 \frac{sin\varphi_{12}}{sin\varphi_{23}} F_x - l_1 cos\varphi_3 \frac{sin\varphi_{12}}{sin\varphi_{23}} F_y \tag{37}$$

$$T_E = l_4 sin\varphi_2 \frac{sin\varphi_{34}}{sin\varphi_{23}} F_x - l_4 cos\varphi_2 \frac{sin\varphi_{34}}{sin\varphi_{23}} F_y \tag{38}$$

2.3 Trajectory Planning Module

For this part, we are going to accomplish a module which is responsible for tracking objects. This is corresponding to the application scenarios where our users can grab a beacon tag, put their luggage on the robot and then the robot can follow their users in the airport.

To create a more user-friendly scenarios (which means our users should take as easy and simple approach as possible to use our product. We finally use a UWB module to provide the position of the users and robots, then use camera if necessary to avoid the obstacles. Then a tracking algorithm is used to give the command to the robot. All the computation work needed will be solved in the host computer since we are afraid of the computing power of the microcomputer.

2.4 Perception Module: UWB positioning technology

Ultra-Wideband (UWB) is a radio communication technology that enables highly accurate tracking by measuring the time it takes for radio waves to travel between devices. This precision, along with its ability to penetrate obstacles and its resilience against signal interference, makes UWB an excellent choice for robotic systems that require reliable real-time location tracking.

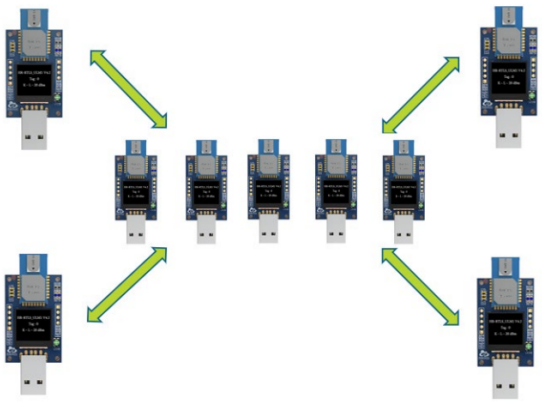


Figure 13: UWB

To set up a positioning system using Ultra-Wideband (UWB) technology, we install three UWB anchors in distinct locations and use two tags that communicate with these anchors. The system calculates the distance from each tag to the anchors based on the time it takes for signals to travel, and then applies triangulation to determine the exact position of the tags in real-time.

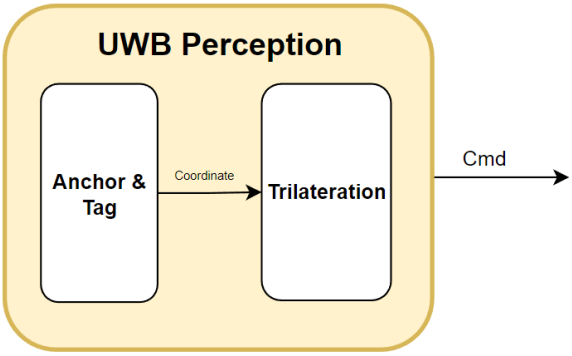


Figure 14: UWB

2.4.1 Trilateration

To solve the trilateration problem using matrices and pseudoinverse, consider three known points A, B, and C with distances d_A, d_B, and d_C from an unknown point P. Linearize the equations by subtracting the squared distances, then express this system in matrix form.

Linear Equations:

$$2(x_B - x_A)x + 2(y_B - y_A)y + 2(z_B - z_A)z = d_A^2 - d_B^2 + x_B^2 - x_A^2 + y_B^2 - y_A^2 + z_B^2 - z_A^2,$$
$$2(x_C - x_A)x + 2(y_C - y_A)y + 2(z_C - z_A)z = d_A^2 - d_C^2 + x_C^2 - x_A^2 + y_C^2 - y_A^2 + z_C^2 - z_A^2.$$

Matrix Form:

$$\mathbf{A} = \begin{bmatrix} 2(x_B - x_A) & 2(y_B - y_A) & 2(z_B - z_A) \\ 2(x_C - x_A) & 2(y_C - y_A) & 2(z_C - z_A) \end{bmatrix}$$

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} d_A^2 - d_B^2 + x_B^2 - x_A^2 + y_B^2 - y_A^2 + z_B^2 - z_A^2 \\ d_A^2 - d_C^2 + x_C^2 - x_A^2 + y_C^2 - y_A^2 + z_C^2 - z_A^2 \end{bmatrix}$$

Pseudoinverse Solution:

$$\mathbf{p} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$$

This approach minimizes the least-squares error in the distances, providing a robust estimate for the coordinates of point P.

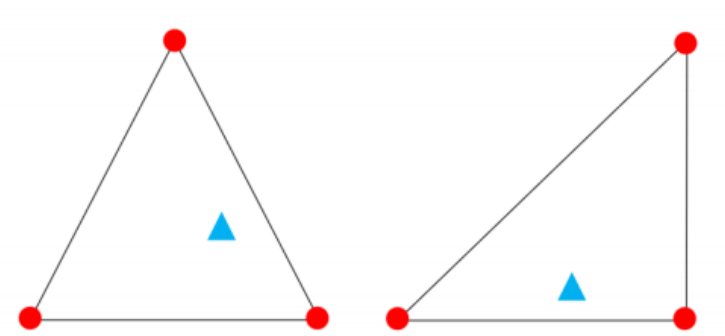


Figure 15: UWBsetting

2.4.2 UWB communication protocol

The serial communication protocol is the data actively uploaded by the UWB module. We define the message protocol as the following, the main content of which is the distance between each label and each anchors. We obtain the position information of each label by using the trilateral measurement method on the host computer

Field	Length	Description
HEAD	mc	Header, indicates 'mc'
USER	00	User ID, default is 00, used to distinguish different users or devices
RANGE0	00000663	Distance to anchor A0, in hex, representing the distance in mm, for example, 1.635m
RANGE1	000005a3	Distance to anchor A1, in hex
RANGE2	00000512	Distance to anchor A2, in hex
RANGE3	000004cb	Distance to anchor A3, in hex
RANGTIME	00146fb7	Measurement time, from the start of transmission to the measurement, in ms

rIdt:IDa	a0:0	Lower byte of ID, anchor ID, tag ID; IDt tag ID short, IDa anchor ID short
END	\r\n	End character

2.5 Tracking algorithm

Given the current position of the robot (x_r, y_r) and its orientation θ_r, and the position of the target object (x_t, y_t), the algorithm first calculates the Euclidean distance d between the robot and the target:

$$d = \sqrt{(x_t - x_r)^2 + (y_t - y_r)^2}$$

The algorithm then calculates the angle θ_{target} from the robot to the target using the arctangent function:

$$\theta_{\text{target}} = \text{atan2}(y_t - y_r, x_t - x_r)$$

The robot must rotate to align its orientation with θ_{target}. If the calculated distance d is greater than the desired following distance d_{follow} plus a buffer (in this case, 20cm), the robot should move forward. If d is less than d_{follow}, the robot should stop or move backward to maintain the desired following distance.

This step is foundational for the tracking algorithm, ensuring that the robot accurately identifies the target direction and estimates the distance to it. This information is crucial for subsequent movement decisions and adjustments.

2.5.1 Velocity Window and Dynamic Window Approach

Integrating DWA, the algorithm considers the robot's dynamic constraints to determine the set of possible velocities (v) and angular velocities (ω), forming the velocity window V_{allowed}.

$$V_{\text{allowed}} = \{(v, \omega) \mid v_{\text{min}} \leq v \leq v_{\text{max}}, \, \omega_{\text{min}} \leq \omega \leq \omega_{\text{max}}\}$$

The Dynamic Window Approach enables the algorithm to optimize velocity selection not only based on the current position and the target's position but also considering the robot's physical capabilities and safety requirements. This flexibility allows the robot to navigate efficiently while adapting to potential obstacles and environmental changes.

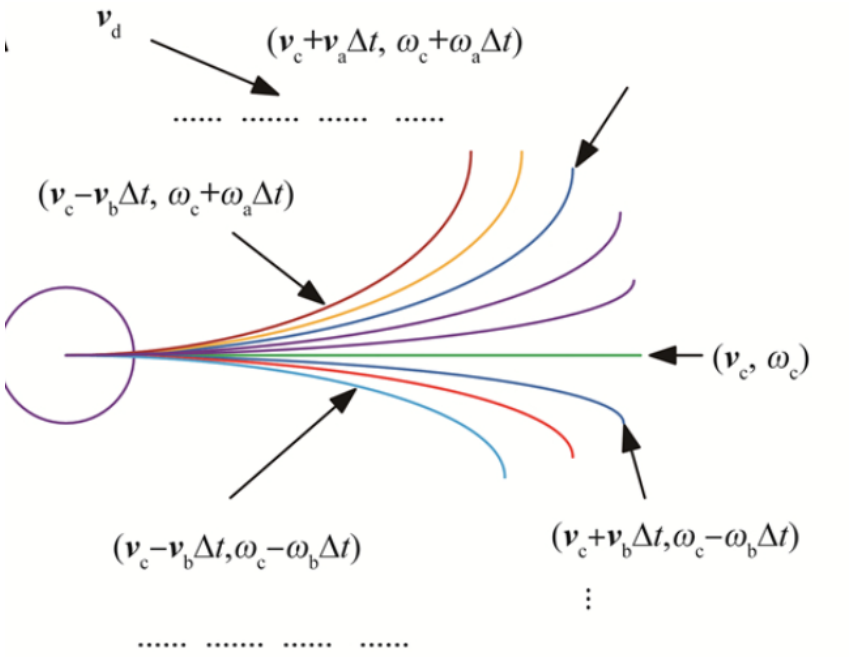


Figure 16: dwa

2.5.2 Velocity Evaluation and Optimal Velocity Selection

The algorithm uses an evaluation function $J(v, \omega)$ to select the optimal velocity combination, taking into account the proximity to the target, the consistency of heading, and the appropriateness of speed.

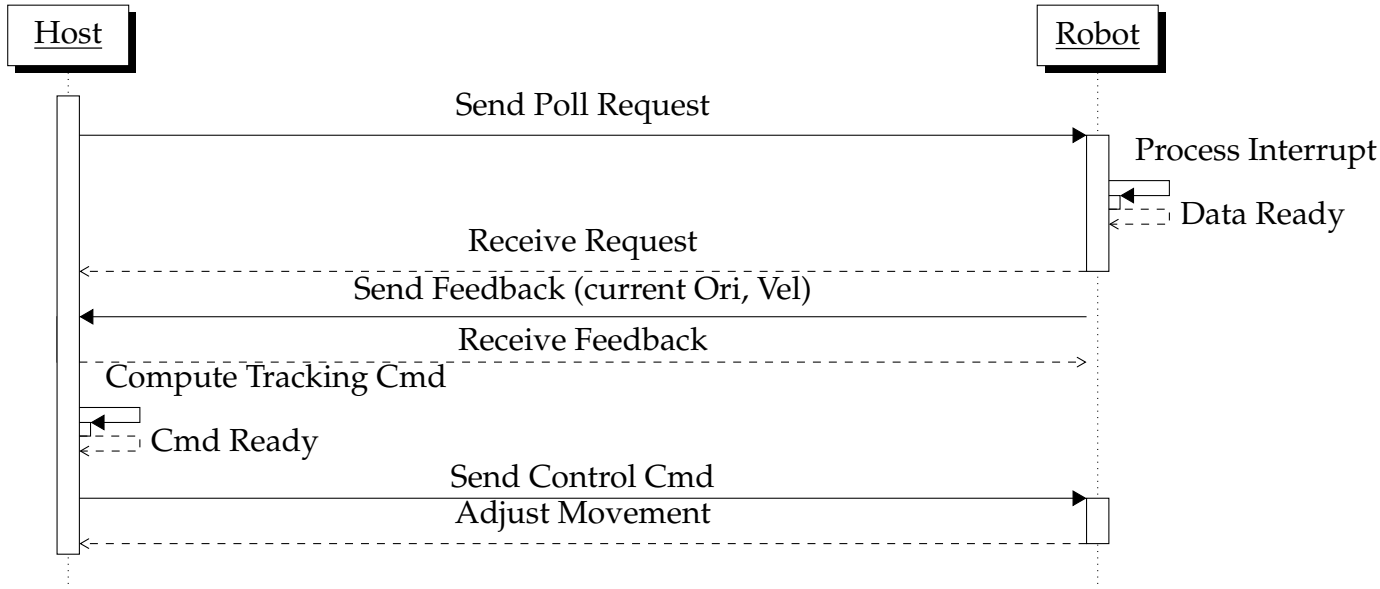
$$J(v, \omega) = \alpha \cdot (d_{\text{target}}) + \beta \cdot \Delta\theta + \gamma \cdot v$$

This function assesses each velocity combination for its effectiveness in approaching the target, heading alignment, and speed appropriateness. By maximizing $J(v, \omega)$, the algorithm selects the most suitable speed and angular velocity at the moment, ensuring efficient, smooth, and safe adjustments to the robot's trajectory towards the target.

2.6 Master/slave communication

To achieve efficient communication between the host and the robot, the choice to use a USB virtual serial port is driven by its advantages in compatibility and transfer speed. The virtual serial port allows the host to communicate data with the robot's microcontroller through a standard USB interface, simplifying the hardware interface requirements while ensuring sufficient transmission rates to meet real-time control needs. Since the host is not mounted directly on the robot, it remotely retrieves the robot's current orientation, position, and speed information, which are crucial for computing the control commands to be sent back to the robot.

Therefore, a complete communication process is established to ensure that data is accurately and timely transmitted between the host and the robot. The following describes each step of the communication process in detail:



The communication process between the host and the robot starts with the host periodically sending Poll requests via a USB virtual serial port to inquire about the robot's current state. This polling serves as a trigger mechanism that alerts the robot to transmit the latest data, including its orientation and speed, back to the host. Upon receipt of each Poll request, the robot processes this input through its USB communication interrupt program, which is crucial for ensuring the robot can respond immediately and prepare the state data for transmission. This immediate processing is vital for maintaining the accuracy and timeliness of the data transfer.

Once the robot has processed the request, it sends the required feedback data back to the host through the same USB virtual serial port. The host then uses this data to compute new control commands based on its built-in algorithms. These commands are specifically tailored to guide the robot's next actions, which may include adjustments to its direction or alterations to its speed to better navigate its environment.

After computing the appropriate commands, the host sends these back to the robot through the USB virtual serial port. Upon receiving these control commands, the robot adjusts its behavior according to the instructions provided, ensuring it operates according to the latest guidance from the host. This continuous loop of communication and control allows for precise management of the robot's operations, ensuring it behaves as expected in its operational context.

The below two tables show the Feedback protocol and the Control Cmd protocol. The poll protocol obeys the same rule, while its three channels are empty, so there is no need to put its protocol here.

As we can see here, the Message mainly composes of four channels, each contains four channels, one has 11 bits, representing one message. This table only includes the data with valid information. For check bit and parity bit and End bit are not included.

In our communication design, everything is done periodically, which means the above process will happen every 0.3 second. Every time when robot get a poll request, it will send his newest info to host for update, and host sends back the new control command for further tracking.

First, for the feedback protocol sent by robot to host for updating information. It consists of three valid info: the current orientation (Yaw angle of robot), the type of message (identify it as the poll msg) and the current speed for DWA algo. In order to make everything easy, we set all the value to be unsigned int. As for ori and speed which may be negative, i apply a offset to make them all positive.

Channel	**Description**	**Values Encoded**	**Bits Used**	**Bit Position Across Bytes**
ch1	Current Speed	Capped at RC_MAX, 1024	11 bits	byte0 [0:7], byte1 [0:2]
ch2	Msg type (Type channel)	Message Type (POLL, Feedback or CMD)	11 bits	byte1 [3:7], byte2 [0:4]
ch3	Current Ori	Yaw Angle * 100 + 36000 [unsigned int for negative]	11 bits	byte2 [5:6], byte3 [0:7], byte4 [0]
ch4	Unused	0	11 bits	byte4 [1:7], byte5 [0:3]

Table 2: Feedback Protocol in robot control communication protocol

Next, for the cmd protocol sent by host to robot for updating control command. It consists of three valid info: the forward speed of two wheels, the type of message (identify it as the cmd msg) and the turning speed to turn around. In order to make everything easy, we set all the value to be unsigned int.

Channel	**Description**	**Values Encoded**	**Bits Used**	**Bit Position Across Bytes**
ch1	Turning speed (or fixed 1024 for `cmd_type` = 1)	Capped at RC_MAX, 1024 if `cmd_type`=1	11 bits	byte0 [0:7], byte1 [0:2]
ch2	Msg type (Type channel)	Message Type (POLL, Feedback or CMD)	11 bits	byte1 [3:7], byte2 [0:4]
ch3	Unused (always set to 0)	Always 0	11 bits	byte2 [5:6], byte3 [0:7], byte4 [0]
ch4	Forward speed (or fixed 1024 for `cmd_type` = 1)	Capped at RC_MAX, 1024 if `cmd_type`=1	11 bits	byte4 [1:7], byte5 [0:3]

Table 3: CMD Protocol in robot control communication protocol

3 Tolerance Analysis

For this project, we use SimuLink to test and observe the variables in our designed control algorithms and use Webots to do the simulation in a physical simulated world.

3.1 SimuLink verification

To assess the stability of our control system, we constructed a path featuring several bars with a radius of 1.5cm, and set the target velocity to 2m/s. Through Simulink, we observed the robot's behavior, and the simulation results are depicted in Figure 18. The observed oscillations in the speed curve, resulting from the obstacles on the path, align precisely with our expectations, demonstrating the resilience of the entire system.

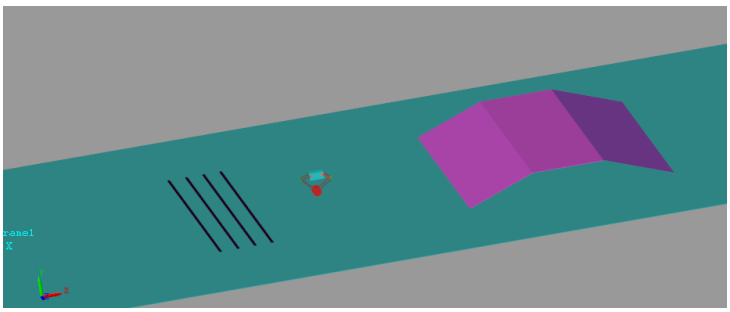


Figure 17: Bar on path

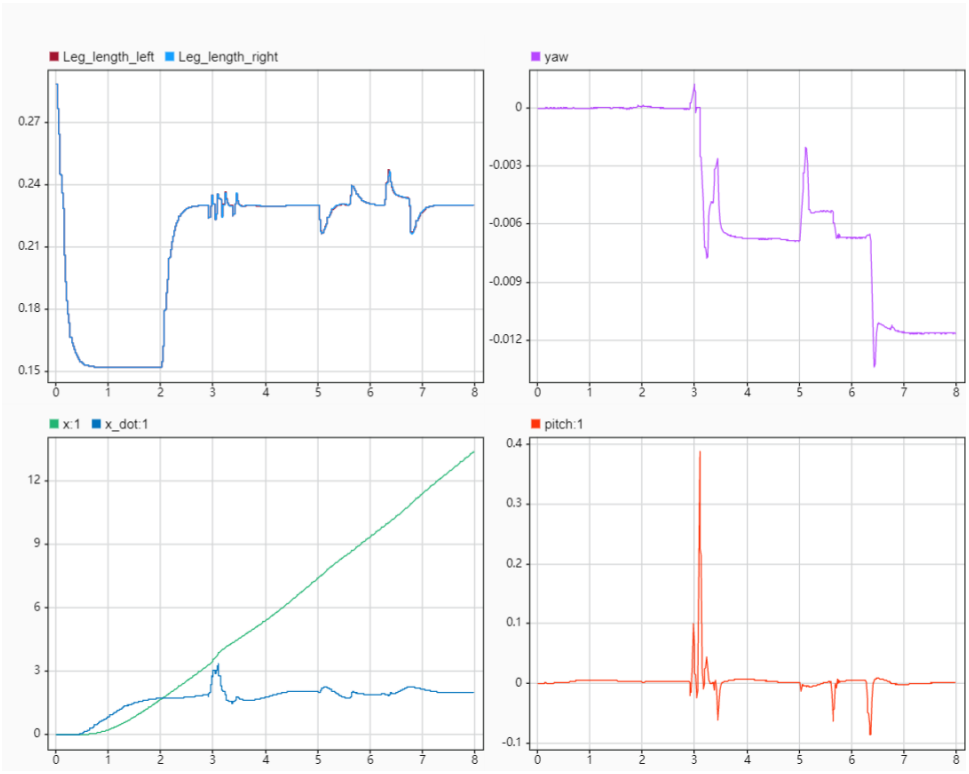


Figure 18: simulation result

3.2 Webots simulation

In this design, we utilized the powerful physics engine and simulation environment provided by Webots, as well as the flexibility and efficiency of the C language, to realize a stable, balanced car system. Our control algorithms are based on Linear Quadratic Regulator (LQR) and Virtual Model Control (VMC), enabling the car to maintain balance under various external disturbances.

To evaluate the feasibility and stability of the design, a tolerance analysis was performed. Through simulation in Webots, we assessed the system's performance, particularly its ability to maintain balance in the presence of external disturbances. The simulation results confirm that the car effectively maintains balance, demonstrating the resilience of the control system. By implementing physics simulation and control algorithms in Webots, we are able to effectively verify the feasibility and stability of the design, optimize control parameters, and ultimately achieve a high-performance balanced car system.

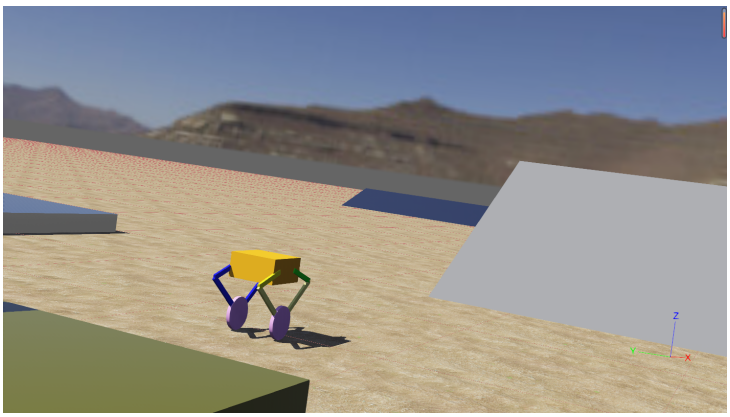


Figure 19: Simulation world in Webots

3.3 Real world testing

The following is the pitch data we measure in real real world testing. The orange line curve is the target pitch angle set by the remote controller, the blue curve is the real pitch read by the IMU unit. We can see the robot responds quite well.

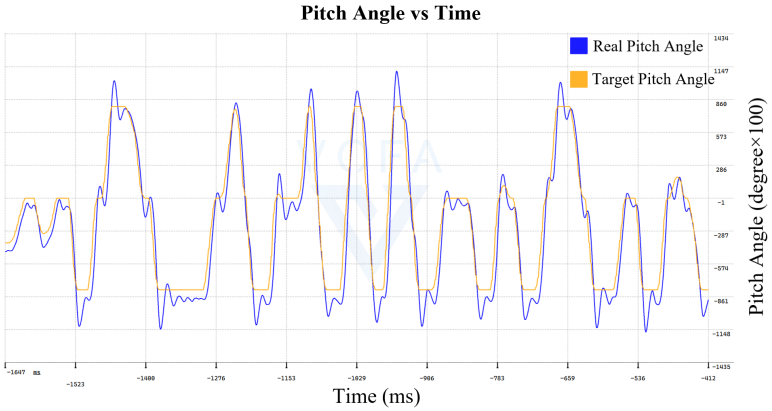


Figure 20: Pitch angle measure without disturbance

4 Cost and Schedule

subsectionCost Analysis The total cost now is 10371 RMB. Four joint motors from Unitree spent 6400 RMB, almost 50 percent of the total cost. The acrylic boards for iteration don't cost as much as we expected, only around 500 RMB.

Purchse Date	Comment	Manufacturer	Quiality	Price (RMB)	
2024.1.18	Unitree A1 Joint Motor	Unitree	4	1600.0	6400
2024.1.19	TTL to 485 module		4	82.5	330
2024.1.20	Bawl set bearing		12	10.4	124.6
	XIAOMI motor	XIAOMI	2	499.5	999
2024.2.20	DaMiao CAN to USB module	DaMiao	1	185.0	185
2024.2.21	Acrylic Parts-1		15	11.3	170
2024.2.25	CNC Parts		10	92.0	920
2024.2.26	KP035 Bearing		2	88.0	176
2024.2.27	Screws		50	1.2	59.81
2024.3.1	Acrylic Parts-2		5	44.0	220
2024.3.7	M3.5 Screws		10	1.9	19
2024.3.7	55×68×7 Bearing		4	19.1	76.56
2024.3.11	DaMiao wires	DaMiao	10	8.8	88
2024.3.12	Unitree RS485 converter	Unitree	1	200.0	200
2024.3.12	RS485 to TTL modeule		1	5.2	5.17
2024.3.12	Nylon 3D print		9	26.7	240
2024.3.19	UART to RS485 module		1	49.0	49
2024.3.23	TTL to RS485 module		2	17.0	34

Figure 21: Bill of material for current robot

4.1 Schedule

Week	Tasks	Person
2024.3.18 - 2024.3.24	CAD modeling	Jiajun Hu
	simulation code test on Webot	Yuhao Wang
	test Aprildetector's performance on single	Yixuan Li
	Test OpenCV algorithms on binocular cam	Xuchen Ding
2024.3.25 - 2024.3.31	Main body assembly	Jiajun Hu
	LQR parameters simulation	Yuhao Wang
	optimze Apriltag's performance and stabili	Yixuan Li
	Optimize CV tracking on binocular camera	Xuchen Ding
2024.4.1 - 2024.4.7	UART to RS485 module test	Jiajun Hu
	parameter measurement of real robot and	Yuhao Wang
	deploy on robot and test performance, mai	Yixuan Li
	deploy on robot and test performance, mai	Xuchen Ding
2024.4.8 - 2024.4.14	Gyroscope algorithm transplantation	Jiajun Hu
	intergate the algorithms with the motor cor	Yuhao Wang
	deploy on robot and test performance, mai	Yixuan Li
	deploy on robot and test performance, mai	Xuchen Ding
2024.4.15 - 2024.4.21	Remote control code transplanatation	Jiajun Hu
	intergate the algorithms with the motor cor	Yuhao Wang
	deploy on robot and test performance, mai	Yixuan Li
	deploy on robot and test performance, mai	Xuchen Ding
2024.4.22 - 2024.4.28	CAD modele iteration	Jiajun Hu
	intergate the algorithms with the motor cor	Yuhao Wang
	overall vision tracking test	Yixuan Li
	overall vision tracking test	Xuchen Ding

Figure 22: Schedule

5 Discussion of Ethics and Safety

5.1 Ethical concern

Our design does not interfere with any life-related experiment or social problem. However, military use of robots or any other misuse of our design that intends to turn the helper robot into a killer is against the IEEE Code of Ethics[3]. Therefore, we promise that we will not open-source the Control System and the Sensor Unit.

5.2 Safety concern

To prevent injuries resulting from collisions with the human body and to address potential experimental accidents during the assembly process, we will implement a collision detection mechanism for the robot. Additionally, an external emergency shutdown button will be installed to prevent any potential hazards. For safety during our robot test, we utilize a stop trigger on our remote control. By pulling this trigger, all the input signal to our robot will be switched to zero and the robot will be automatically shut down.

References

[1] V. Klemm, A. Morra, C. Salzmann, *et al.*, "Ascento: A two-wheeled jumping robot," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 7515–7521. DOI: 10.1109/ICRA.2019.8793792.

[2] Z. L. CHEN Yang WANG Hongxi, "Control of wheel-legged balancing robot[j]," in *INFORMATION AND CONTROL*, vol. 52, 2023, pp. 648–659. DOI: 10.13976/j.cnki.xk.2023.2533.

[3] IEEE. ""IEEE Code of Ethics"." (2016), [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (visited on 02/08/2020).

Appendix A Denotation of the variables

A.1 Variable explanation

Table 4: Variables used in our model

Variable	Explanation	Unit
M_w	The mass of each wheel	kg
M	The mass of the whole robot	kg
r	The radius of each wheel	m
D	The distance between the left and right wheel	m
x_r, x_l	The horizontal displacement of the wheel	m
w_r, w_l	The angular velocity of the wheel	rad/s
g	The gravity constant	N/kg
T_r, T_l	The torque of the wheel	N*m
F_r, F_l	The horizontal friction force by the ground	N
H_r, H_l	The horizontal force on the wheel by the motor	N
V_r, V_l	The vertical force on the wheel	N
θ	The pitch angle of the body rotated around z-axis	rad
ϕ	The yaw angle of the body rotated around z-axis	rad
I	The moment of inertia of the wheel	kg*m^2
J	The moment of inertia of the body rotated around z-axis	kg*m^2
J_ϕ	The moment of inertia of the body rotated around y-axis	kg*m^2
L	The distance from the center of mass of the body to the z-axis	m
D_{joint}	The distance between the two joint motors	m

A.2 Measurement of the variables

Table 5: Variables used in our model

Variable	value	Unit
M_w	0.5	kg
M	8.341	kg
r	0.06225	m
L	0.1225	m
D_{joint}	0.15	m