ECE 445

SENIOR DESIGN LABORATORY

DESIGN DOCUMENT

Design Document 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.

2 Design

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.

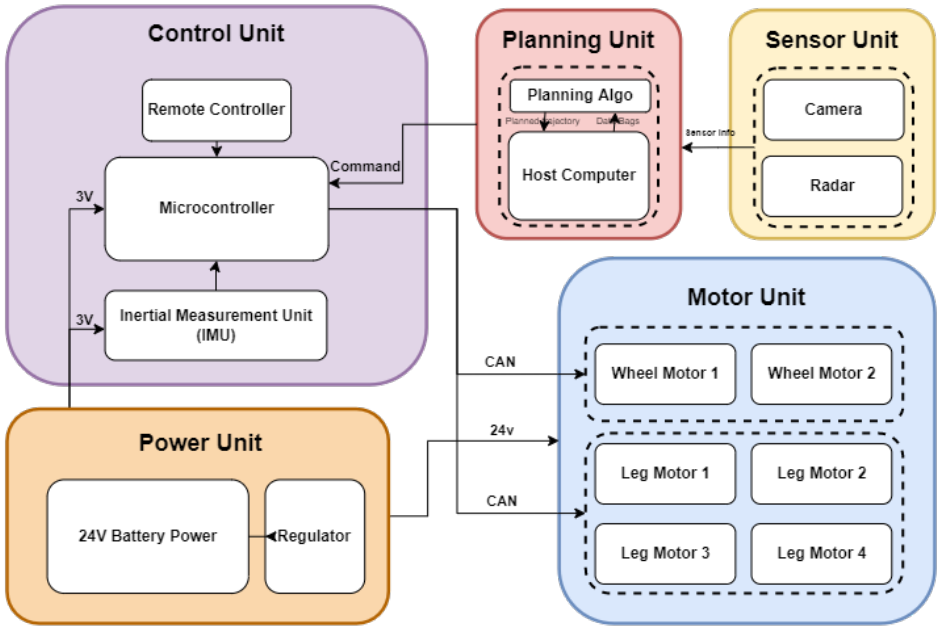

Figure 2: Top Level

2.1 Mechanical Structure

For the design of the balancing robot, we mainly consider aspects such as size and stability. We determined that the overall size of the robot needs to be around 500mm×500mm, so it can carry a normal-sized handbag. In terms of stability, we mainly focus on the leg joints. We found that 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.

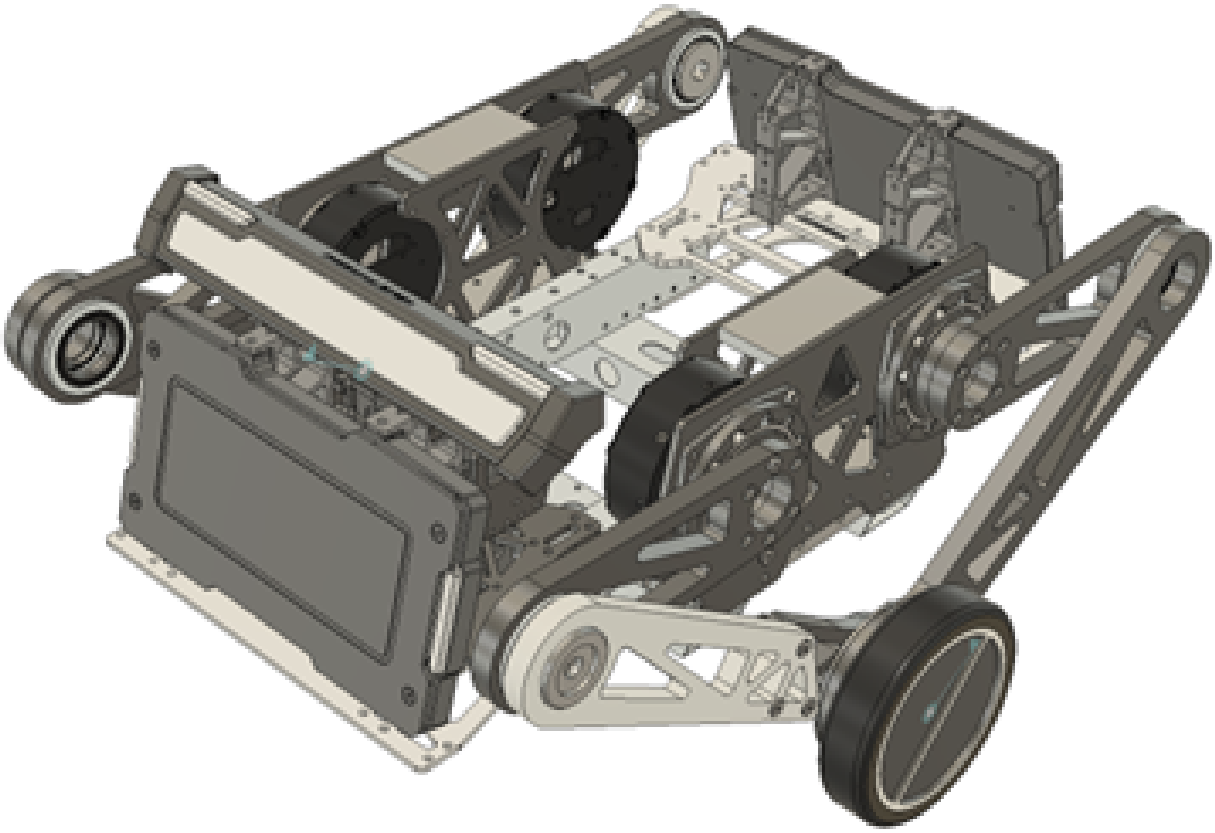

Figure 3: Overview for prototype 1

2.2 Mechanical System

We assume the weight of our balance robot is 20kg. At the hip joint, we use Unitree Technology's A1 joint motor, which has a peak torque of 33.5 Nm, meeting our requirements for joint motors.

At the leg joints, we use CNC machined aluminum parts and bowl group bearings for joint fixation. This method of fixation, compared to using screws, allows for a larger joint shaft diameter, effectively reducing the wobble in the leg joints. It also minimizes the parts number we need.

At the wheel joints, we use Xiaomi's Cyber Gear motors, along with KP035 robot joint bearings to share the load of the wheel motors.

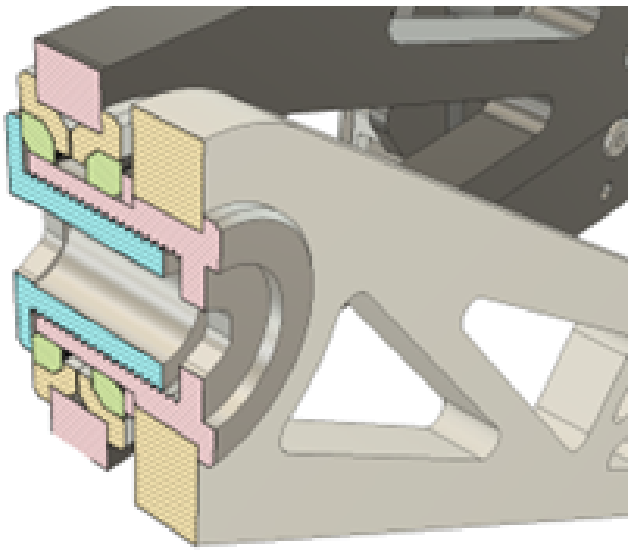

Figure 4: Leg joint cross-section view

In terms of materials, our final robot uses carbon fiber materials because they have a smaller mass compared to other materials at the same strength. We also adopt 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.

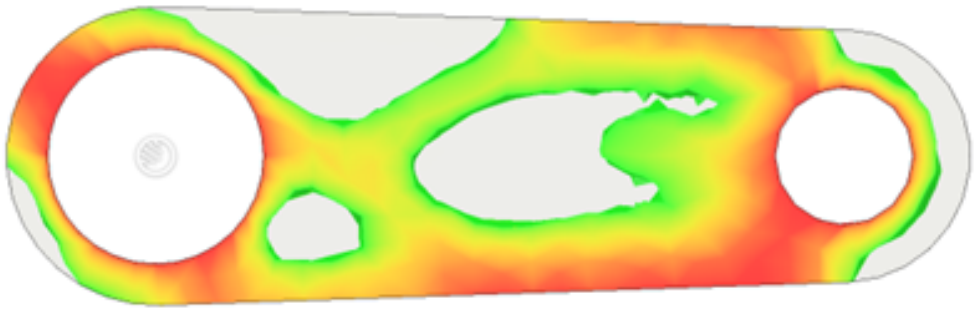

Figure 5: Upper leg topology result

This is the picture of the current robot legs. We are testing our design first using acrylic boards due to their low cost. In the final version we plan to use carbon fiber boards for lighter weight and higher material strength.

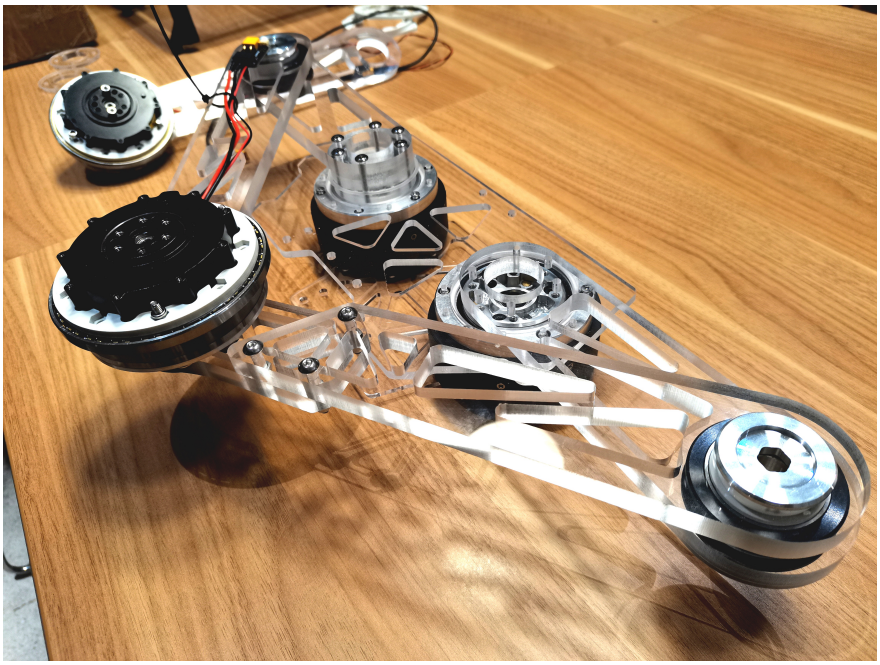

Figure 6: Robot Leg

2.3 Electrical System

2.3.1 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.

2.3.2 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.

2.4 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.

2.5 Control System

As for the control system, since our robot's controlling strategy mainly consists of two parts: the wheel part and the leg part. As suggested in [1], we plan to use the Linear quadratic regulator (LQR) algorithm for both of them and see what performance looks like. And this process will be tested on matlab simulink and Webot for simulation and later on moved to the real robot.
The forward motion of the car can be decomposed into forward motion and relative rotation (pitch) around the center of mass P of the car body.

2.5.1 Wheel model

The force analysis diagram for the left and right wheels of the two-wheeled robot is shown in the figure. The resultant force in the horizontal direction for the robot is the vector sum of the frictional force between the wheels and the ground, as well as the horizontal force 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:

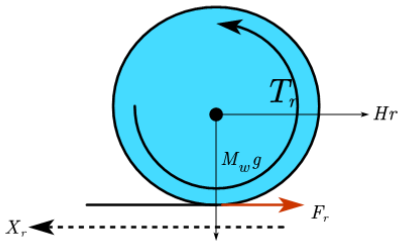

Figure 7: Wheel model

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 $\omega_r = \frac{\dot{x_r}}{r}$, 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 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 that:

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

2.5.2 Moving forward and backward

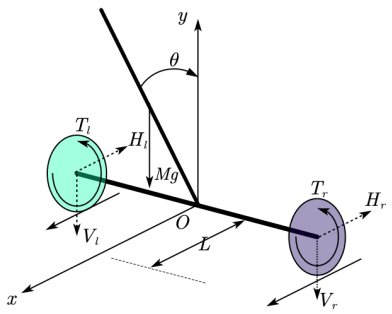

Figure 8: Moving forward and backward

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

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

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:

$$J\ddot{\theta} + T_l + T_r = (V_l + V_r)Lsin\theta - (H_l + H_r)Lcos\theta \tag{8}$$

Combining (6), (7) and (8), 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:

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

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

2.5.3 Rotation Motion

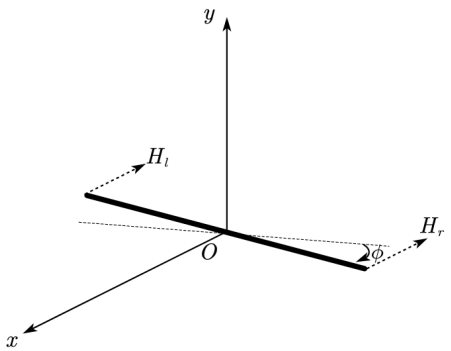

Figure 9: Rotation model

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 equation (3) into the equations above, resulting in:

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

2.5.4 LQR control system

With equation (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,

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:

$$J = \frac{1}{2} \int_0^\infty (X^T Q X + u^T R u) dt \tag{20}$$

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.
In order to select the most appropriate set of Q and R, we planned to use MATLAB to simulate the performance of different choices.

2.6 Host System Trajectory Planning Module

For this part, we plan to move all the computation work to the host computer since we are afraid of the computing power of the microcomputer. Of course, if we latter proved the microcomputer can cater to this work, we will abandon this unit. Till now, we still want to use the ROS operating system to let the robot pass rosbags to the host system, which contains all the pos and vel infomation of the robot. What's more, the camera and radar unit will also catch the corresponding information. Then we will use an open source following algorithm to plan the trajectory and give specific command for robot to execute.(e.g: move forward, backward ....)

2.7 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.

To set up a positioning system using Ultra-Wideband (UWB) technology, 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.

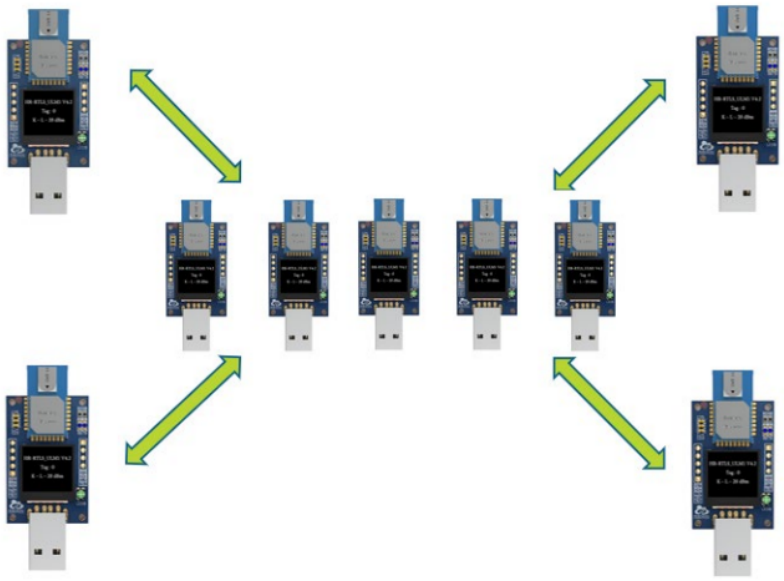

Figure 10: UWB

2.7.1 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.

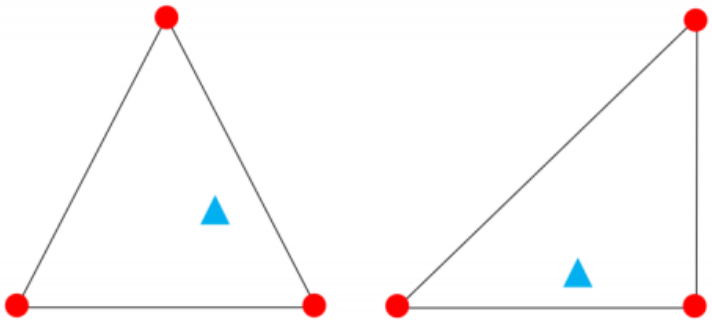

Figure 11: UWBsetting

2.7.2 Serial 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

2.7.3 other embedded software on UWB module

This part is provided by the UWB module manufacturer, I do not write this part except modification on protocol mentioned above.

2.7.4 test

Currently, I have tested the module of protocol part to get the distances of each tag, the results is as follows.

rangetime	tag_ID	x(m)	y(m)	z(m)	range0(m)	range1(m)	range2(m)	range3(m)	range4(m)	range5(m)	range6(m)	range7(m)	rx_pwr(dBm)
106925	Tag 0				null	null	0.17	null	null	null	null	null	-78.19
106970	Tag 0				null	null	0.17	null	null	null	null	null	-78.19
107037	Tag 0				null	null	0.16	null	null	null	null	null	-78.53
107082	Tag 0				null	null	0.16	null	null	null	null	null	-78.53
107149	Tag 0				null	null	0.16	null	null	null	null	null	-78.68
107194	Tag 0				null	null	0.16	null	null	null	null	null	-78.68
107261	Tag 0				null	null	0.17	null	null	null	null	null	-78.65
107306	Tag 0				null	null	0.17	null	null	null	null	null	-78.65
107373	Tag 0				null	null	0.17	null	null	null	null	null	-78.38
107418	Tag 0				null	null	0.17	null	null	null	null	null	-78.38
107485	Tag 0				null	null	0.18	null	null	null	null	null	-78.44
107530	Tag 0				null	null	0.18	null	null	null	null	null	-78.44
107597	Tag 0				null	null	0.18	null	null	null	null	null	-78.18
107642	Tag 0				null	null	0.18	null	null	null	null	null	-78.18
107709	Tag 0				null	null	0.18	null	null	null	null	null	-78.15
107754	Tag 0				null	null	0.18	null	null	null	null	null	-78.15
107821	Tag 0				null	null	0.18	null	null	null	null	null	-78.43
107866	Tag 0				null	null	0.18	null	null	null	null	null	-78.43
107933	Tag 0				null	null	0.18	null	null	null	null	null	-78.63
107978	Tag 0				null	null	0.18	null	null	null	null	null	-78.63
108045	Tag 0				null	null	0.18	null	null	null	null	null	-78.35
108090	Tag 0				null	null	0.18	null	null	null	null	null	-78.35
108157	Tag 0				null	null	0.17	null	null	null	null	null	-78.29
108202	Tag 0				null	null	0.17	null	null	null	null	null	-78.29
108269	Tag 0				null	null	0.17	null	null	null	null	null	-78.53
108314	Tag 0				null	null	0.17	null	null	null	null	null	-78.53
108381	Tag 0				null	null	0.17	null	null	null	null	null	-78.61
108426	Tag 0				null	null	0.17	null	null	null	null	null	-78.61
108493	Tag 0				null	null	0.17	null	null	null	null	null	-78.45

Figure 12: test

As for the Trilateration part, I adopt a efficiency algorithm and it is proved to be accurate.

2.8 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.

After that, we need to convey the information we need back to the robot, here, we decide to use DJI remote control protocol. The detailed protocol is as follows.

Index	**Position X**	**Position Y**	**Position Z**	**Control Flag**	**Status Flag**
Length	48	64	80	96	104
Length	16	16	16	8	8
Valid	Yes	Yes	Yes	No	Yes
Range	Max 32767	Max 32767	Max 32767	Max 1	Max 1
Range	Min -32768	Min -32768	Min -32768	Min 0	Min 0
Default	0	0	0	0	0
Description	Movement along X, Y, Z axes			0 - no action; 1 - action	
Description	Bit0 - W, Bit1 - S, Bit2 - A, Bit3 - D				
Description	Bit4 - Q, Bit5 - E, Bit6 - Shift, Bit7 - Ctrl				

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 14. 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.

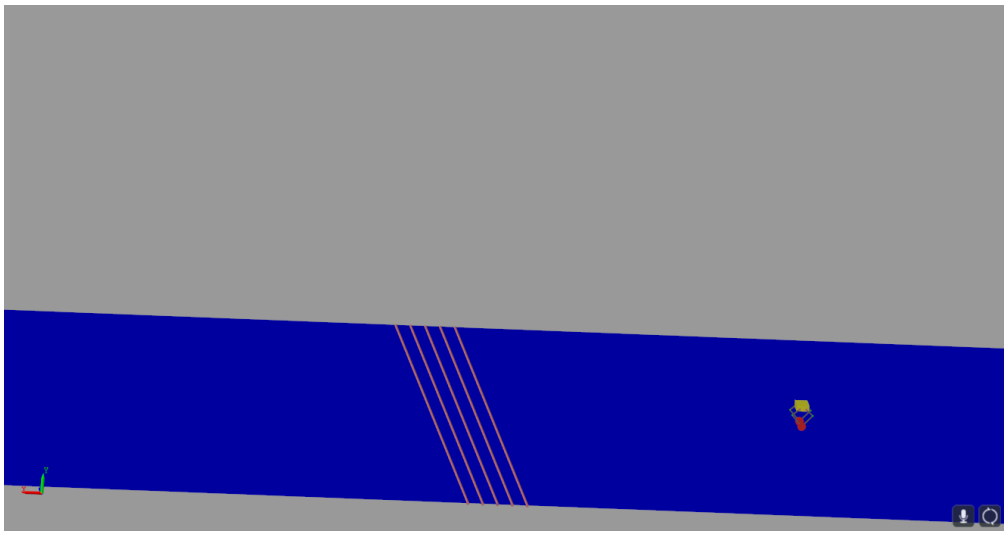

Figure 13: Bar on path

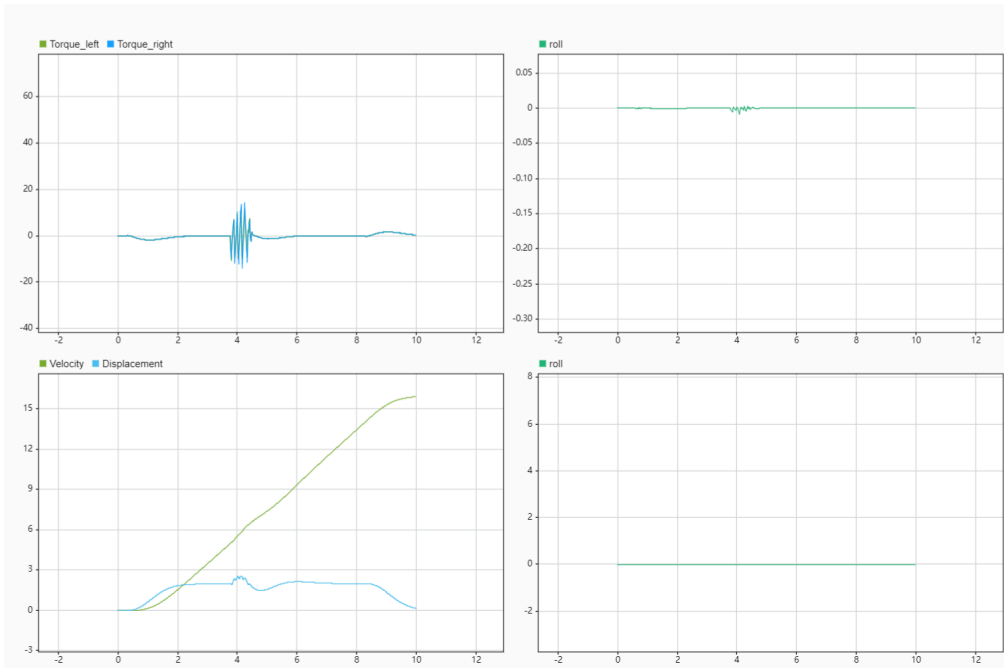

Figure 14: simulation result

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.

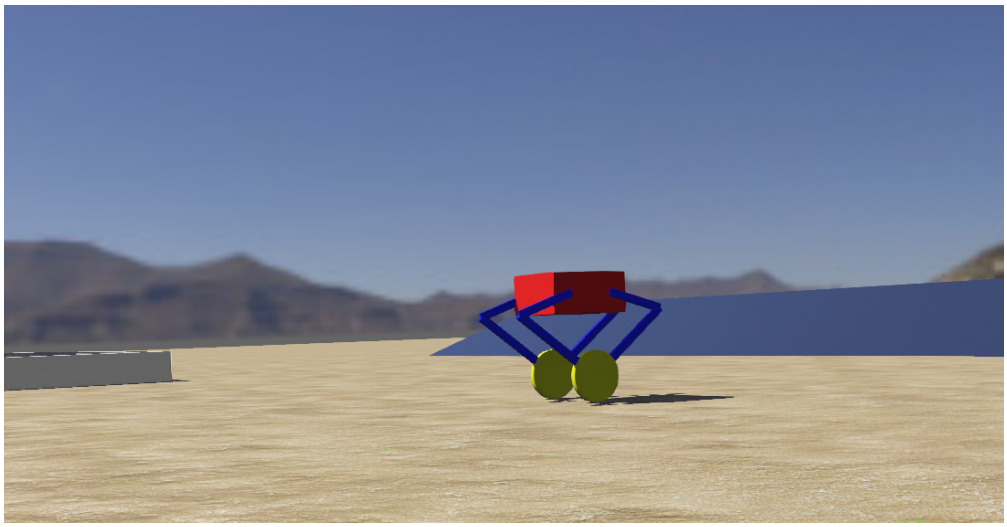

Figure 15: Simulation world in Webots

4 Cost and Schedule

4.1 Cost 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 16: Bill of material for current robot

4.2 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 stabilit	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, mal	Yixuan Li
	deploy on robot and test performance, mal	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, mal	Yixuan Li
	deploy on robot and test performance, mal	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, mal	Yixuan Li
	deploy on robot and test performance, mal	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 17: 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[2]. 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] 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 3: 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 4: 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