ACEBOTT

Robot Arm Car Tutorial

Perface

Our Company

ACEBOTT STEM Education Tech Co.,Ltd

Founded in China's Silicon Valley in 2013, ACEBOTT is a STEM education solution leader We have a team of 150 individuals, including members from research and development, sales, and logistics. Our goal is to provide high-quality STEM education products and services to our customers. We are working together with STEM education experts and our business partners to produce successful STE products together Our self-own factory also provides CEM services for our clients including Logo customization on product packaging and PCB.

Our Tutorial

The Robot Arm Car kit is an expansion kit for the QD001 smart car. This tutorial is based on the foundation of the QD001 smart car. It is recommended that you familiarize yourself with the QD001 smart car tutorial before starting this tutorial. The purpose of this tutorial and the Robot Arm Car kit is to expand and extend the functions of the smart car, as well as to help users deepen their understanding of Robot Arm Car. If you want a Robot Arm Car that can move freely, this kit and tutorial will provide you with knowledge and operational guidance to help you build and control your own Robot Arm Car.

Through this kit, you can:

1. Learn how to effectively use the ESP32 controller board, including downloading code, understanding its features, and coding in the Arduino IDE.
2. Build a solid programming foundation through a blockly coding language, as the ESP32 employs such a language to control circuits and sensors.
3. Understand the structure and control principles of Robot Arm, and learn servo motor control to coordinate the Robot Arm with the car to complete integrated tasks.
4. Step-by-step build your own Robot Arm Car using the ACEBOTT kit according to the tutorial, enhancing your maker skills.

5. Implement basic functions such as web control and App control in the Robot Arm Car project.
6. Enhance a comprehensive understanding of the concepts of Robot Arm Car, preparing for future advanced learning.

Customer service

ACEBOTT is a dynamic and fast-growing STEM education technology company that strives to offer excellent products and quality services that meet your expectations. We value your feedback and encourage you to drop us a line at support@acebott.com with any comments or suggestions you may have.

Our experienced engineers are dedicated to promptly addressing any problems or questions you may have about our products. We guarantee a response within 24 hours during business days.

Follow Us

Scan the QR codes to Follow Us for troubleshooting & the latest news.
We have a very large community that is very helpful for troubleshooting and we also have a support team at the ready to answer any questions.

ACEBOTT FB Group QR Code

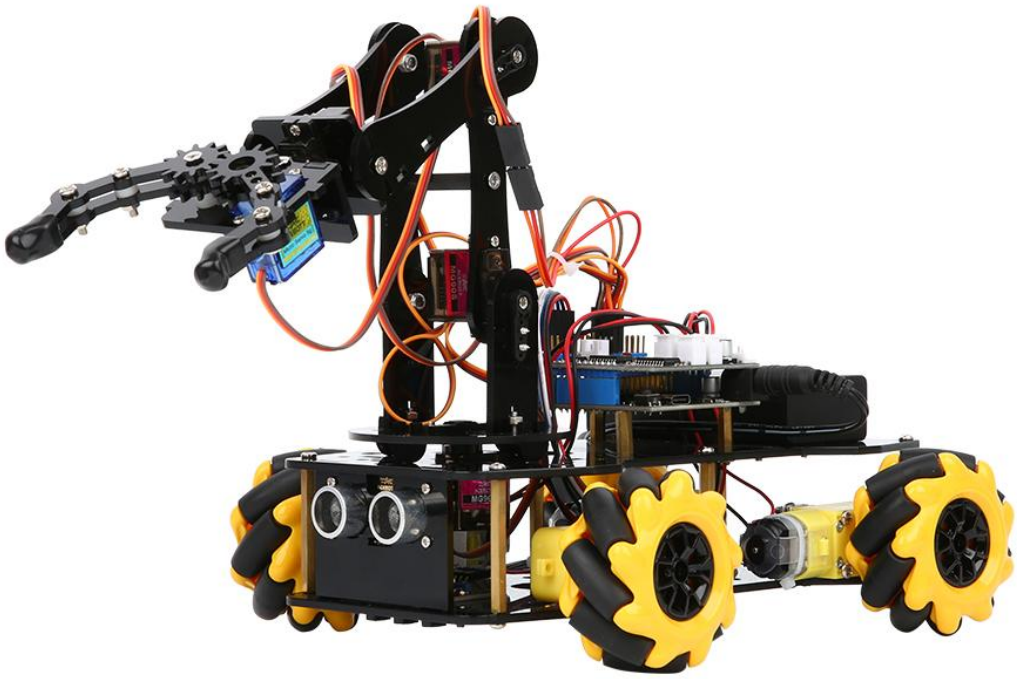
YouTube QR Code

Contents

Lesson 1 Robot Arm Motion Control

Robot Arm Car is a complex intelligent machine that integrates a robotic arm with a mobile platform, capable of freely moving within specific spaces to grasp and transport objects. They exhibit high flexibility and adaptability, enabling them to accomplish various diverse tasks.

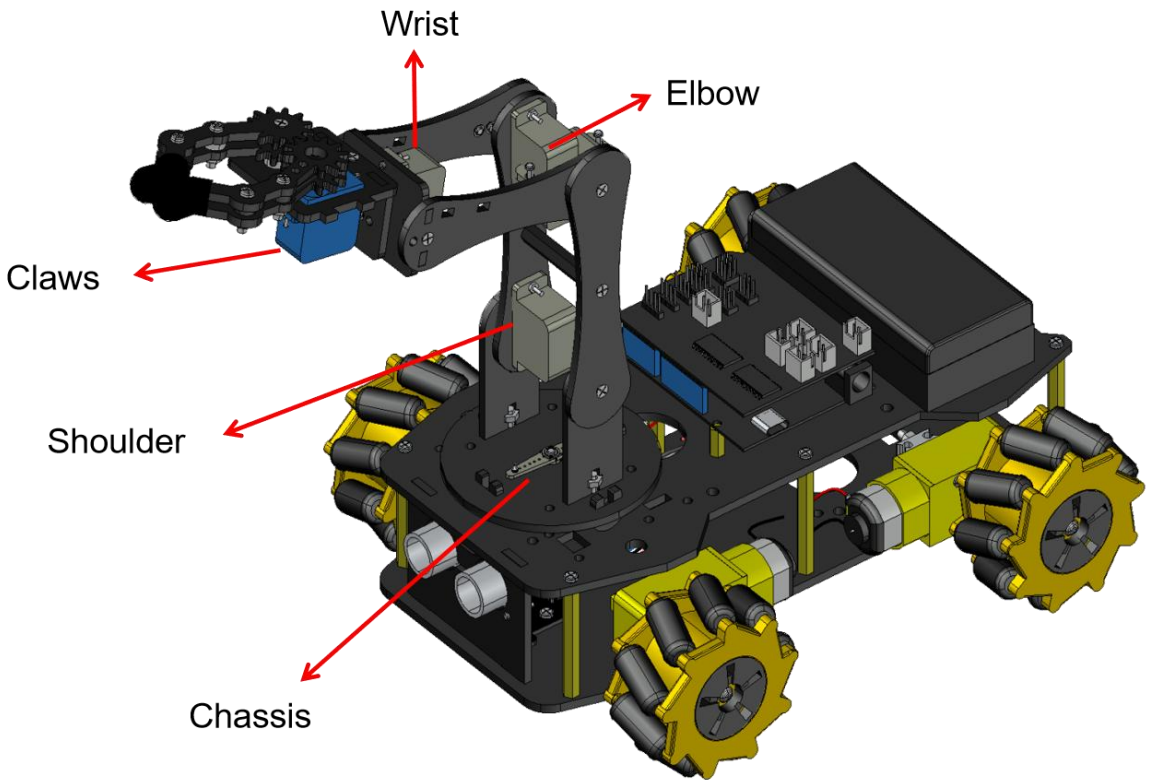
Robot Arm Cars have extensive applications in daily life. In industrial production, they can move along assembly lines to perform tasks such as welding, tightening, and assembly. They also automate material handling and distribution, thereby enhancing production efficiency. In the military sector, Robot Arm Cars can be deployed for tasks such as search and rescue operations. In agriculture, they enable automated harvesting of fruits and vegetables.



This tutorial's Robot Arm Car consists of three main parts: the chassis of the QD001 smart car, the robot arm, and the ESP32 controller. Detailed explanations for the QD001 mobile chassis and ESP32 controller are provided in the QD001 tutorial. Here, the focus is specifically on the robot arm components.

Ⅰ. Robot Arm Structure

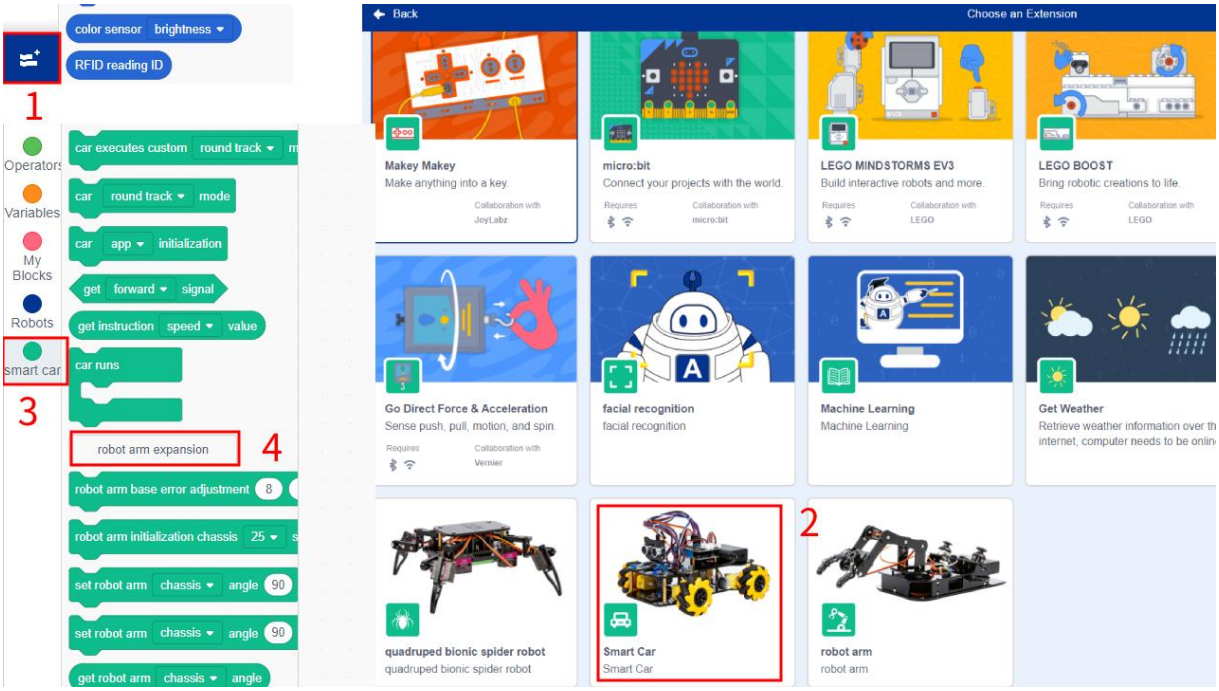
A robot arm is a mechanical structure composed of several joints and links, where these links are tightly interconnected through precise joints to work together and accurately simulate the flexible movements of a human arm. The motion of a robot arm relies on the movement of its joints, which are driven by motors mounted on each joint. Generally, the greater the number of independently driven joints, the more flexible the robot arm. Based on the number of independently driven joints, robot arm structures are categorized by their degrees of freedom. A robot arm with 3 independently driven joints is known as a 3-axis robot arm or 3-degree-of-freedom (DOF) robot arm, while one with 4 independently driven joints is referred to as a 4-axis robot arm or 4-degree-of-freedom (DOF) robot arm, and so forth.



The robot arm used in this tutorial is a 5-axis arm, consisting of a base, shoulder, elbow, wrist, and end effector. Each joint in the robot arm is driven by precise servo motors to ensure smooth and accurate movements. Specifically, the first axis is controlled by the base servo motor for rotation, the second axis by the shoulder servo motor, the third axis by the elbow servo motor, the fourth axis by the wrist servo motor

for rotation, and the fifth axis by the end effector servo motor. This design allows the robot arm to perform complex and varied tasks while improving its precision and efficiency.

For the control of the robot car, you need to add the "Smart Car" extension. Click the "Add Extension" button in the lower left corner of ACECode to add the "Smart Car" extension in the pop-up page. In addition to the extension instructions of the smart car, this extension also contains the operation instructions related to the expansion package of the smart car.



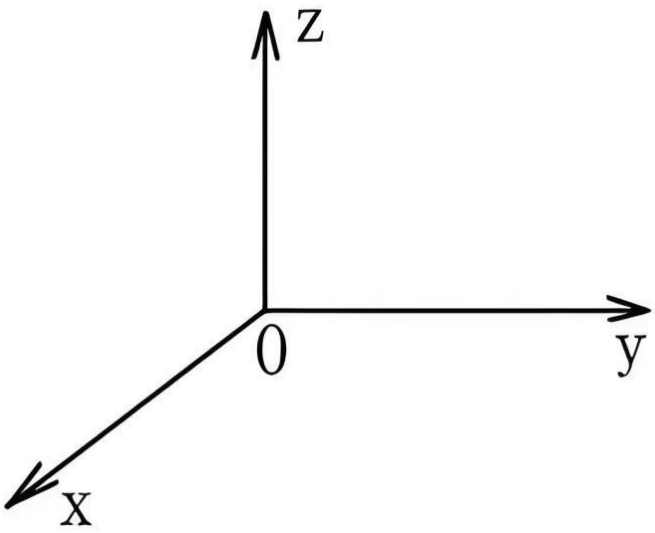
Note: 1. This tutorial is applicable to ACECode version 2.0 and above. You can check the software version number in the upper left corner of the ACECode software. Please make sure that the software version you are using meets the requirements;2. If you need to update the ACECode software version, you can go to the ACEBOTT official website: https://www.acebott.com/pages/software to download the latest ACECode software version.

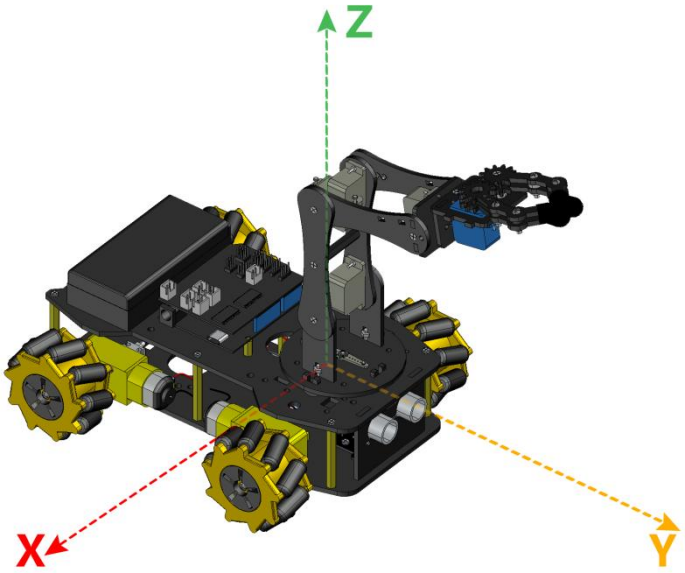
The motion control of a robot arm involves controlling the movement of motors at each joint to position the end effector of the robot arm at a specified location, and then executing corresponding tasks. To accurately control the robot arm's end effector to reach a target position, we need to know the coordinate information of the target position. There are many ways to represent the coordinates of the robot arm's end

effector. Here, we primarily discuss how to represent the position of the robot arm's end effector in Cartesian coordinates and joint coordinates.

Ⅱ. Cartesian Coordinate System

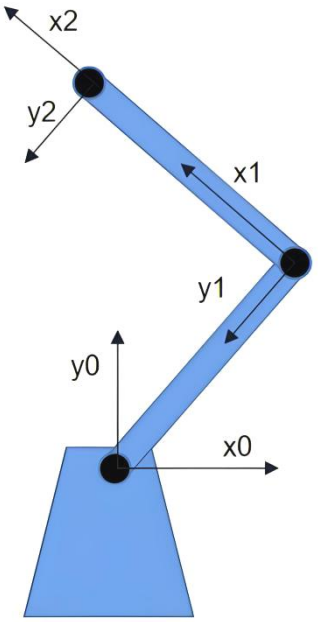
In the context of motion control for robot arm vehicles, the Cartesian coordinate system is one of the most commonly used coordinate systems. It is a mathematical system used to describe the position of points in space. In three-dimensional space, the Cartesian coordinate system consists of three mutually perpendicular axes (x, y, z). The intersection of these three axes is the origin (O) of the coordinate system. In this tutorial, the origin of the Cartesian coordinate system is located at the center of the Chassis servo disc of the robot arm.





Ⅲ. Joint Coordinate System

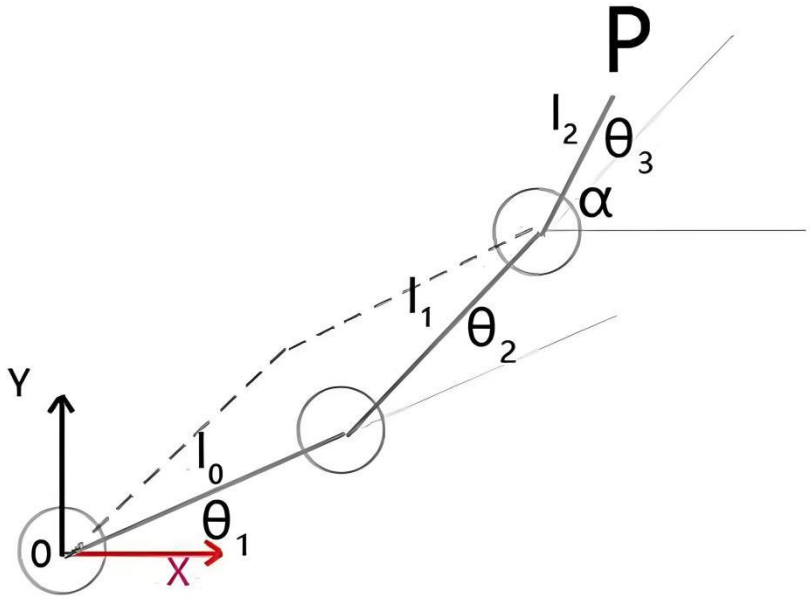
In addition to the Cartesian coordinate system, each joint of the robot arm vehicle has its own coordinate system known as the joint coordinate system. Its origin is typically located at the joint connection point, and its axes are defined along the axis of rotation of the joint. Each joint coordinate system has a joint coordinate associated with it, which describes the rotation angle or extension of that joint. Since each joint of the robot arm vehicle can rotate or extend, the joint coordinate systems can be transformed according to the current pose of the robot arm vehicle.



Ⅳ. Forward And Inverse Kinematics

Forward kinematics refers to calculating the position and orientation of a robot arm vehicle in Cartesian coordinates based on joint angles. Inverse kinematics, on the other hand, involves calculating the joint angles necessary to achieve the motion of the robot arm vehicle based on Cartesian coordinates.

Forward kinematics problems can be solved using matrix transformations. This involves comparing the transformation matrices of each joint coordinate system with the coordinate transformation formula of the robot arm vehicle to determine the angles of each joint. Inverse kinematics problems, however, are typically more complex, requiring the solution of a set of nonlinear equations and often involving multiple possible solutions.



In this tutorial, we primarily establish a Cartesian coordinate system with the origin at the center of the Chassis servo disc. When a specified position coordinate (X, Y, Z) is inputted, the tutorial's library automatically invokes an inverse kinematics algorithm to calculate the coordinates of each joint of the robot arm. These coordinates are then converted into angles for each servo motor, thereby controlling the robot arm's end effector to reach the target point in spatial coordinates.

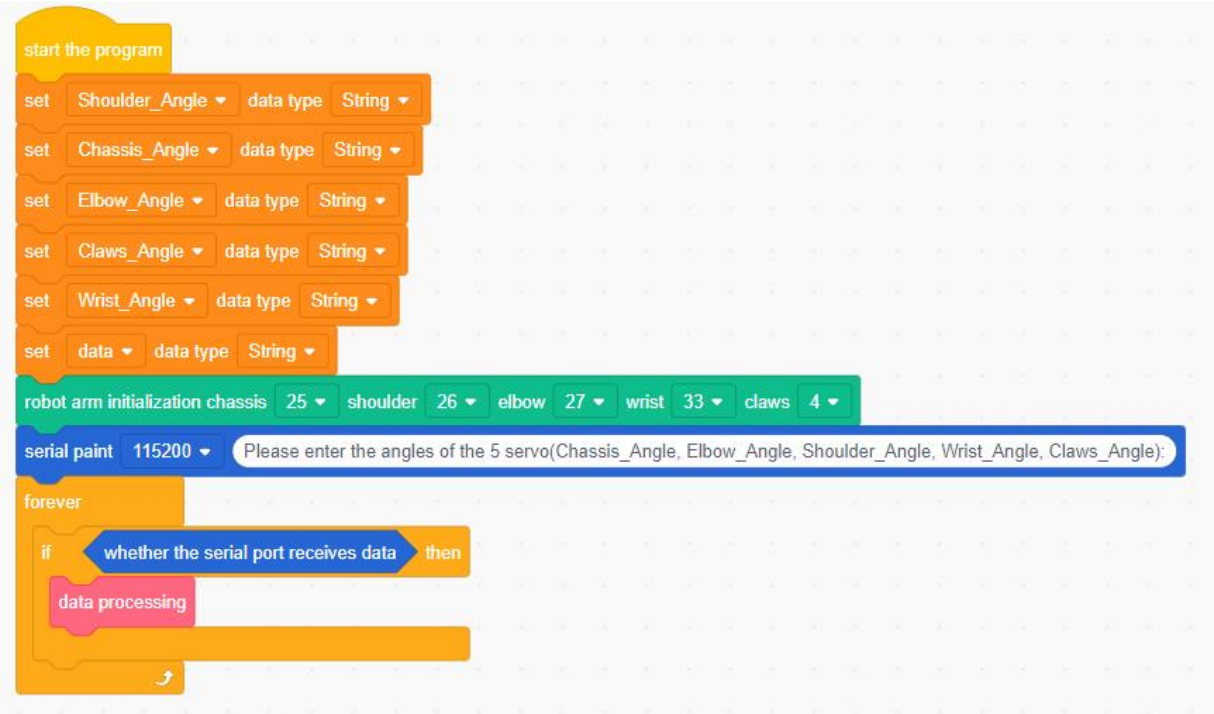
Ⅴ. Robot Arm Joint Control

Joint control of the robotic arm involves controlling the angle of each servo motor in the joint coordinate system to position the end effector of the robotic arm at the target location. Next, we will use a program to perform joint control of the robotic arm. Input the angles of five servos in the serial monitor to move the robotic arm's end effector to the corresponding positions.

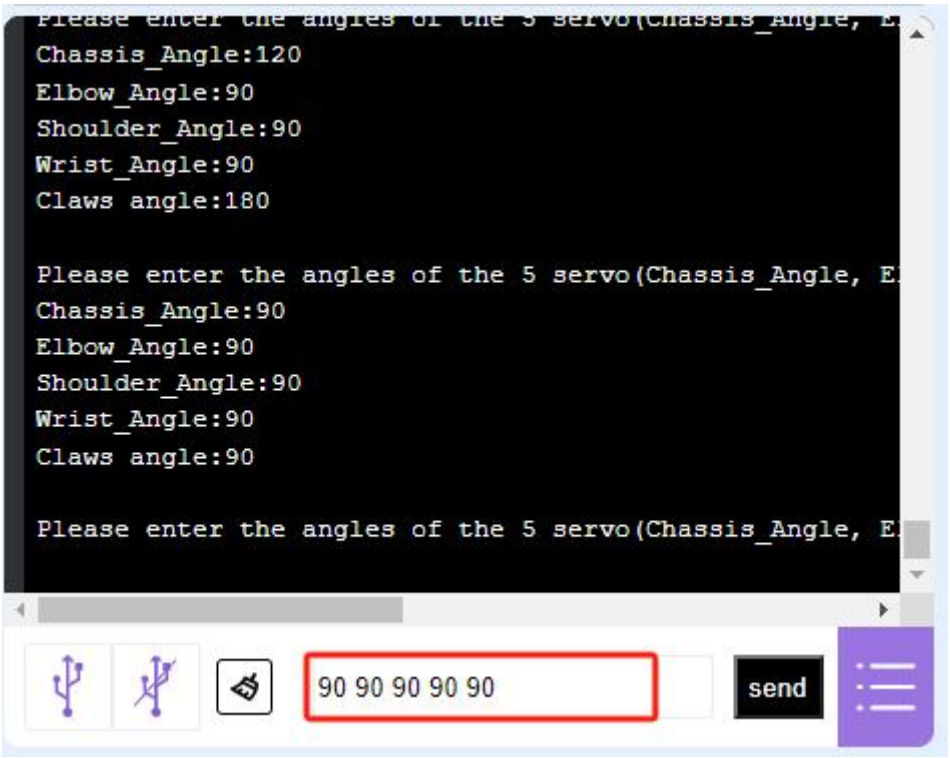
Open "Joint_Control.sb3" in "English\ACECode(Blockly Code)\3.Program\lesson1" ,connect the ESP32 controller board to the computer using a USB cable. Ensure the correct controller board model and port number are selected. Upload the code to the ESP32 controller board. The controller board should be powere

d by the battery pack, with the battery pack switch set to the "ON" position.

Sample Code:




define data processing
set data to receive serial port data
set Chassis_Angle to split data by and take item 1
set Elbow_Angle to split data by and take item 2
set Shoulder_Angle to split data by and take item 3
set Wrist_Angle to split data by and take item 4
set Claws_Angle to split data by and take item 5
serial paint 115200 join Chassis_Angle: Chassis_Angle
serial paint 115200 join Elbow_Angle: Elbow_Angle
serial paint 115200 join Shoulder_Angle: Shoulder_Angle
serial paint 115200 join Wrist_Angle: Wrist_Angle
serial paint 115200 join Claws angle: Claws_Angle
set robot arm chassis angle convert Chassis_Angle to integer
set robot arm elbow angle convert Elbow_Angle to integer
set robot arm shoulder angle convert Shoulder_Angle to integer
set robot arm wrist angle convert Wrist_Angle to integer
set robot arm claws angle convert Claws_Angle to integer
wait 0.5 seconds
serial paint 115200 Please enter the angles of the 5 servo(Chassis_Angle, Elbow_Angle, Shoulder_Angle, Wrist_Angle, Claws_Angle):

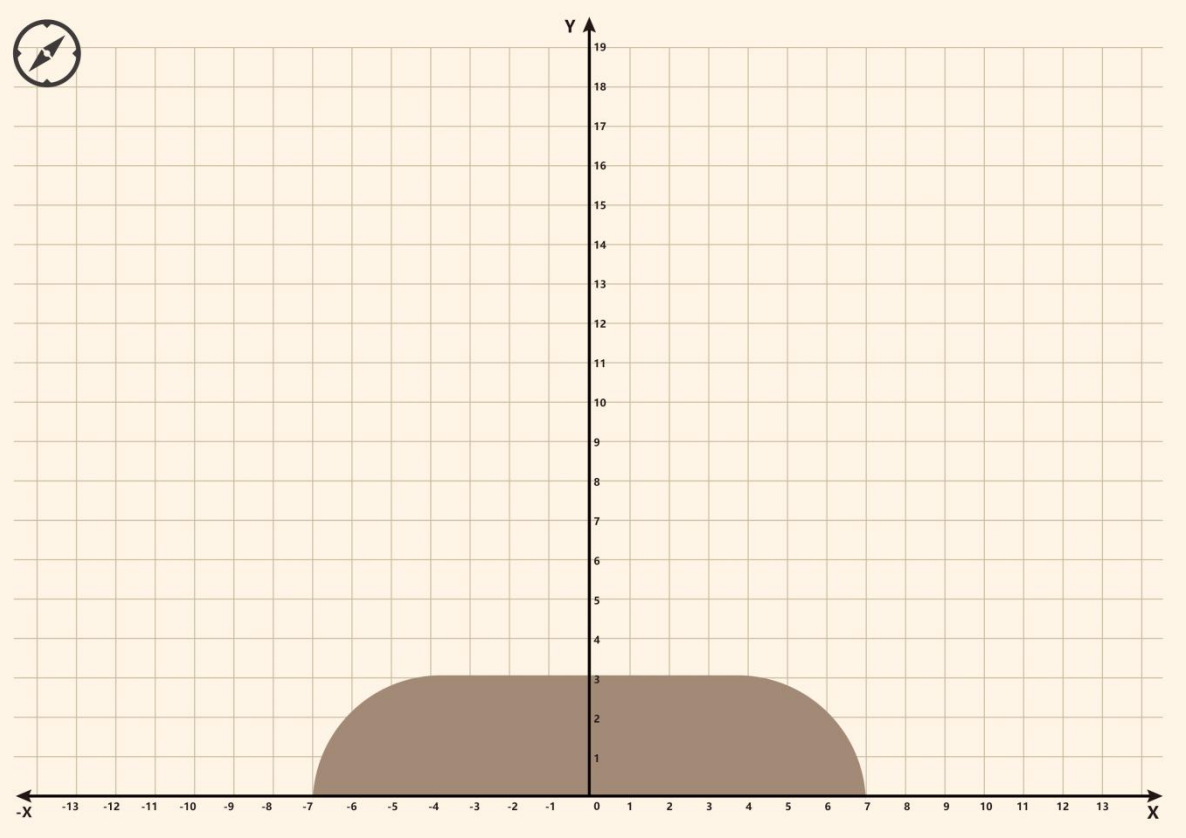


After uploading the program, input the angles corresponding to the five servos in the serial monitor. These 5 angle values should be separated by spaces. Once you have entered them, press “send” button. You will see the robot arm adjust its pose according to the input angles.

Ⅵ. Space Coordinate Control

1. Space coordinate diagram of robot arm

There will inevitably be some errors in the assembly process of the robot arm. In order to better calibrate the robot arm, we need to use the coordinate diagram.The coordinate diagram of the robot arm is composed of X and Y coordinates, with the intersection point of X and Y being the origin of the map. The coordinate range of X is [-13, 13], and the coordinate range of Y is [0, 19]. The chamfered rectangle in the shaded area is the reference position for the robot arm. Place the upper edge of the robot arm's base against this shape.

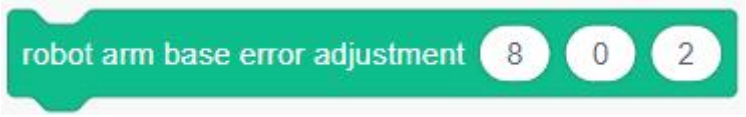


Click to get the robot arm coordinate diagram PDF file

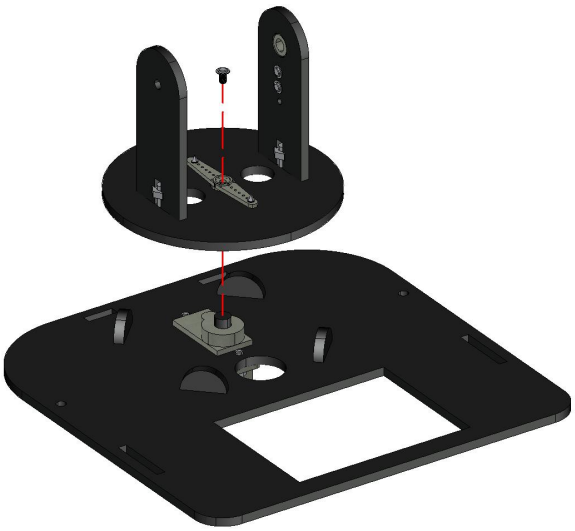
Attention: Please print the diagram of the robot arm on A4 paper according to the PDF file.

2. Robot arm space coordinate error calibration

The robot arm is composed of multiple servos, and in practical work, there may be certain errors. To reduce these errors, we have added instructions in the robot arm control program that allow adjustment of the arm's errors.



The first parameter in this command is used to adjust the angle deviation of the Chassis steering gear. If the Chassis actual angle is less than the servo input angle, enter a value greater than 0 to adjust the deviation. If the actual angle of the Chassis is greater than the steering gear input angle. Enter a value less than 0 to adjust the bias.



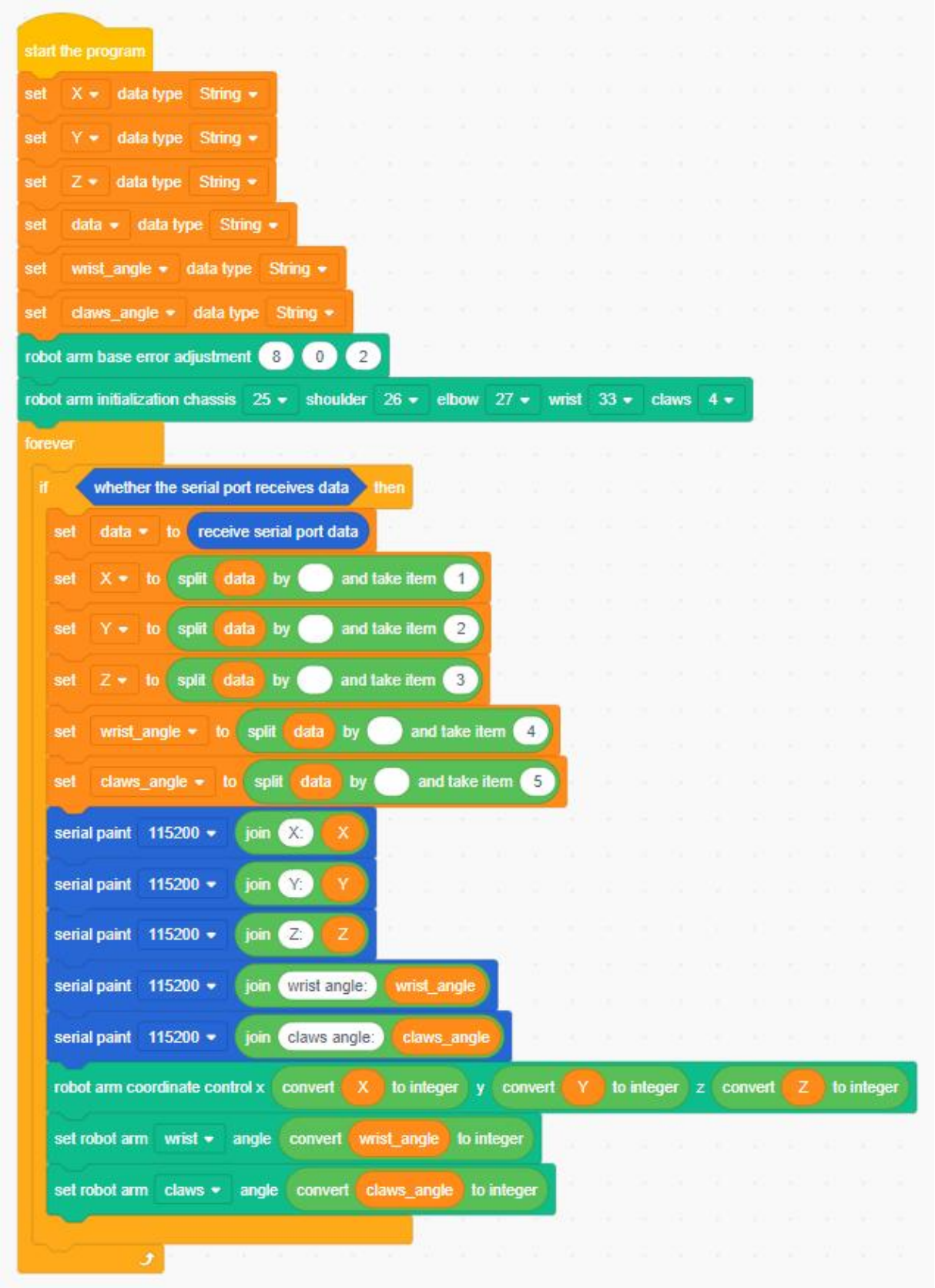
If the error of the first parameter is adjusted, the robot arm still has error, then use the second and third parameters to fine-tune, the default values of these two parameters are 0, the second parameter is when the end of the robot arm is in the positive half axis of X, if the end is still deflected 1 degree to the right, write 1 directly, if it is deflected 1 degree to the left, write-1, if it is not deflected, write 0.

The third parameter is that when the end of the robot arm is in the negative half axis of X, if the end is still deflected by 1 degree to the right, it will be written directly 1, if it is deflected by 1 degree to the left, it will be written-1, if it is not deflected, it will be written 0.

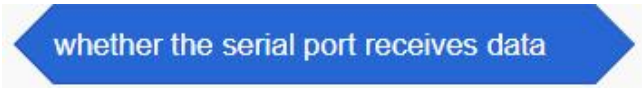
3. Robot arm coordinate control program

Open "Inverse_Kinematics.sb3" in "English\ACECode(Blockly Code)\3.Program\lesson1" ,connect the ESP32 controller board to the computer using a USB cable. Ensure the correct controller board model and port number are selected. Upload the code to the ESP32 controller board. The controller board should be powered by the battery pack, with the battery pack switch set to the "ON" position.

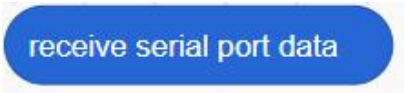
Sample Code:


start the program
set X data type String
set Y data type String
set Z data type String
set data data type String
set wrist_angle data type String
set claws_angle data type String
robot arm base error adjustment 8 0 2
robot arm initialization chassis 25 shoulder 26 elbow 27 wrist 33 claws 4
forever
if whether the serial port receives data then
set data to receive serial port data
set X to split data by and take item 1
set Y to split data by and take item 2
set Z to split data by and take item 3
set wrist_angle to split data by and take item 4
set claws_angle to split data by and take item 5
serial paint 115200 join X: X
serial paint 115200 join Y: Y
serial paint 115200 join Z: Z
serial paint 115200 join wrist angle: wrist_angle
serial paint 115200 join claws angle: claws_angle
robot arm coordinate control x convert X to integer y convert Y to integer z convert Z to integer
set robot arm wrist angle convert wrist_angle to integer
set robot arm claws angle convert claws_angle to integer

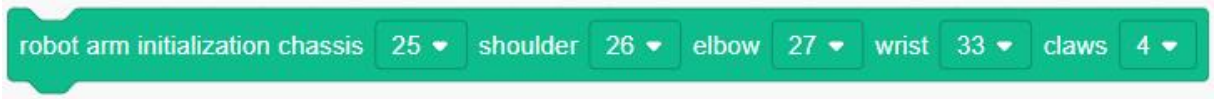
After uploading the program, enter five values in the serial monitor: the X coordinate, the Y coordinate, the Z coordinate, the wrist angle and the claw opening angle (ranging from 90 degrees to 180 degrees). These five values need to be separated by spaces. After entering the values, press “send”button.



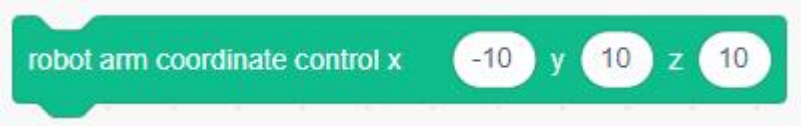
command in the "Robots" category, it can determine whether the serial port has data input.



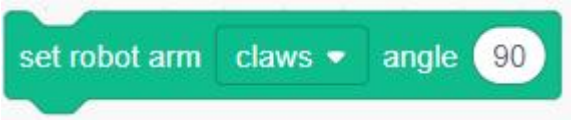
command is in the "Robots" command module, through which the serial port input data can be obtained.



Is the robot initialization command, use it to set the pins of each joint of the robot arm, and let each joint enter the initialization state.



is a robot arm coordinate control command, through which the robot arm end is controlled to move to the specified position in space coordinates, where x is x-axis coordinate, y is y-axis coordinate, and z is z-axis coordinate.



is an arm angle control command, by which the angle of each joint can be controlled.

If the coordinates are correct, the robot arm moves to the spatial coordinate point. If the coordinates are input, the serial port appears "Out of range!" The prompt is because the range of motion of the robot arm is in the sphere, so there will be input values beyond the position that the robot arm can reach. At this time, you need to adjust the data and then re-enter. If the input paw angle is not in the range of 90-180, it will automatically calibrate to a value close to 90-180. If the input paw angle is 50, it will automatically calibrate to 90.

Lesson 2 Roadblock Clearing Of Robot Arm Car

In today's era of rapid technological advancement, robot arms have become an indispensable part of modern industry, commercial services, and everyday life across various fields. With their astonishing flexibility, precision, and efficiency, they have revolutionized traditional operational methods.

When equipped with a mobile chassis, robot arms' capabilities are further expanded and enhanced. For instance, we can utilize robot arm car for tasks such as transportation on roads, assisting in rescue operations, and clearing obstacles. These advancements undeniably greatly facilitate human work and daily life.

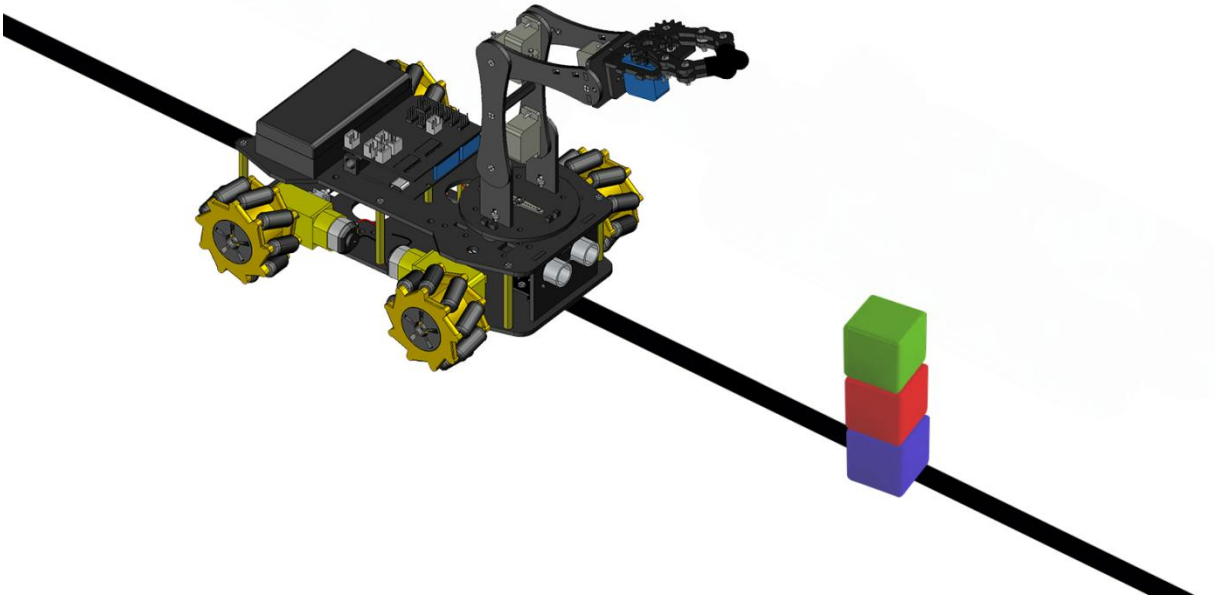
Therefore, in this lesson, we will learn how to use a robotic arm car to simulate obstacle clearing on a line-following road.

Ⅰ. The Robot Arm Car Roadblock Cleaning Procedure

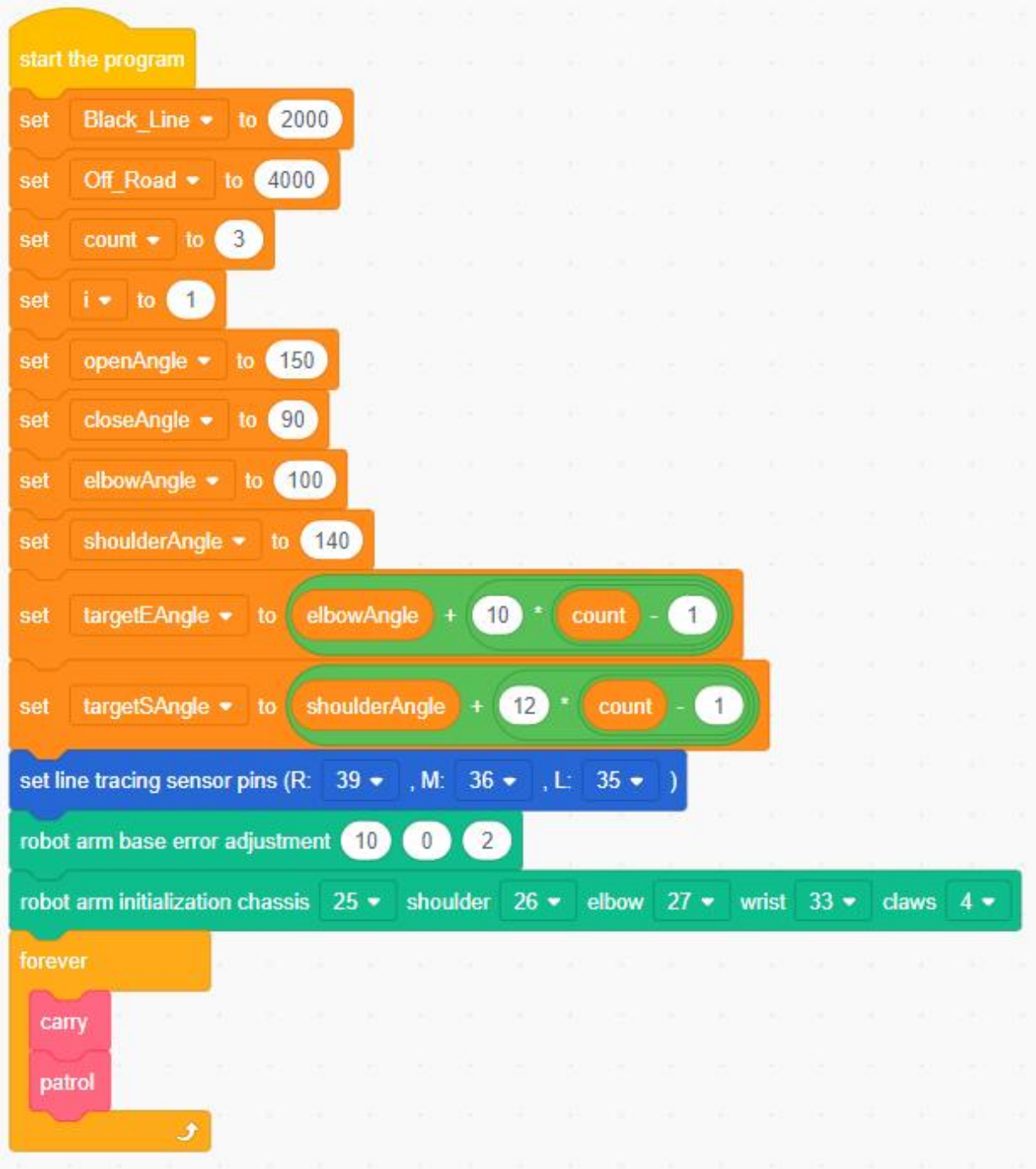
Open “Robot_Arm_Car_Stacking.sb3” in “English\ACECode(Blockly Code)\3.Program\lesson2”,connect the ESP32 controller board to the computer using a USB cable. Ensure the correct controller board model and port number are selected. Upload the code to the ESP32 controller board. The controller board should be powered by the battery pack, with the battery pack switch set to the "ON" position.

Using the black tape from the QD001 kit, create a straight line on the ground as a line-following track. Stack three objects vertically and place them on the path of the line-following track. Position the vehicle on the line-following track with its bottom line sensors facing the path.

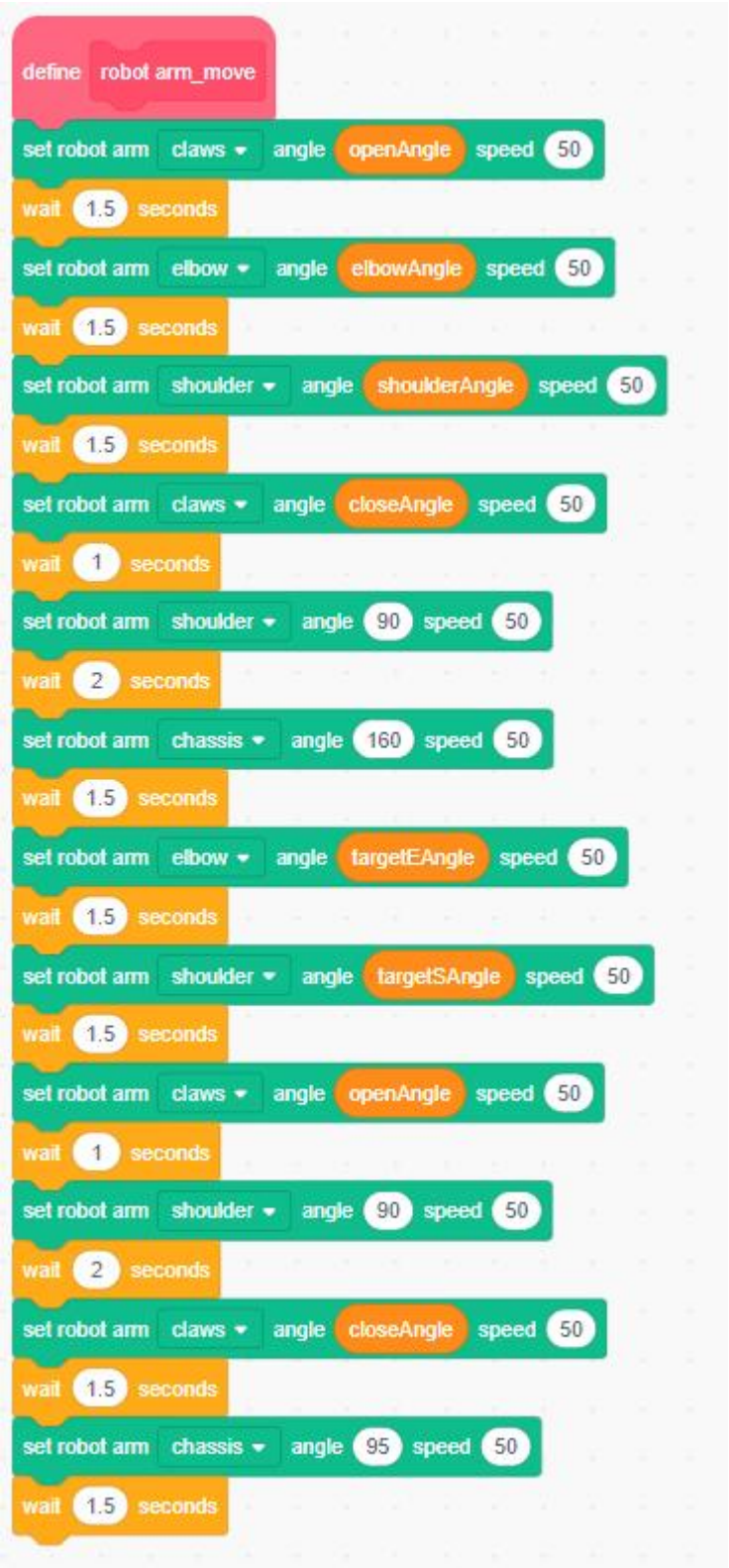
Attention: Please stick a straight-line line-following track, as curved tracks may cause deviations in obstacle detection for the robot arm car.



Sample Code:

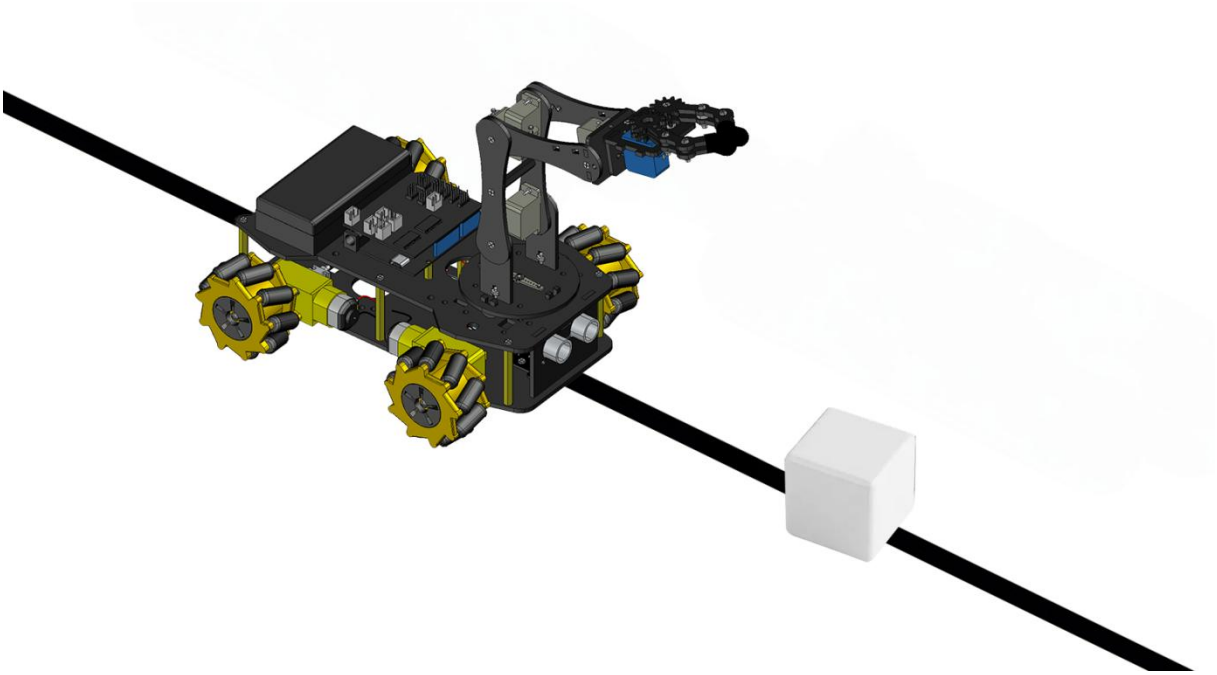

start the program
set Black_Line to 2000
set Off_Road to 4000
set count to 3
set i to 1
set openAngle to 150
set closeAngle to 90
set elbowAngle to 100
set shoulderAngle to 140
set targetEAngle to elbowAngle + 10 * count - 1
set targetSAngle to shoulderAngle + 12 * count - 1
set line tracing sensor pins (R: 39 , M: 36 , L: 35)
robot arm base error adjustment 10 0 2
robot arm initialization chassis 25 shoulder 26 elbow 27 wrist 33 claws 4
forever
carry
patrol


define carry
set UT_distance to ultrasonic(cm) trig 13 echo 14
serial paint 115200 UT_distance
if UT_distance = 15 then
car stops
set robot arm shoulder angle 90 speed 30
wait 1 seconds
set robot arm elbow angle 90 speed 30
wait 2 seconds
repeat until i > count
robot arm_move
if count > 1 then
set elbowAngle to elbowAngle + 10
set shoulderAngle to shoulderAngle + 12
set targetEAngle to targetEAngle - 10
set targetSAngle to targetSAngle - 12
change i by 1
set i to 1
set elbowAngle to elbowAngle - 10 * count - 1
set shoulderAngle to shoulderAngle - 12 * count - 1
set targetEAngle to targetEAngle + 10 * count - 1
set targetSAngle to targetSAngle + 12 * count - 1
robot arm reset


define robot arm_move
set robot arm claws angle openAngle speed 50
wait 1.5 seconds
set robot arm elbow angle elbowAngle speed 50
wait 1.5 seconds
set robot arm shoulder angle shoulderAngle speed 50
wait 1.5 seconds
set robot arm claws angle closeAngle speed 50
wait 1 seconds
set robot arm shoulder angle 90 speed 50
wait 2 seconds
set robot arm chassis angle 160 speed 50
wait 1.5 seconds
set robot arm elbow angle targetEAngle speed 50
wait 1.5 seconds
set robot arm shoulder angle targetSAngle speed 50
wait 1.5 seconds
set robot arm claws angle openAngle speed 50
wait 1 seconds
set robot arm shoulder angle 90 speed 50
wait 2 seconds
set robot arm claws angle closeAngle speed 50
wait 1.5 seconds
set robot arm chassis angle 95 speed 50
wait 1.5 seconds

After uploading the program, we observe that the robot arm car initially autonomously follows the line-following track. When it encounters an obstacle ahead, it stops moving forward. Then, the robot arm controls the end effector to reach the initial coordinates of the obstacle block, grasps the uppermost block, and transports it to the left side of the track for placement. Afterward, it returns to the position of the block and repeats this action. This process continues until all obstacle blocks are cleared. Once cleared, the robot arm car resumes its line-following movement.

Attention: Because the final initial pose of the robot arm in each person's assembled robot arm car is not completely consistent, there will be some errors in the robot arm car. Especially when the robot arm is picking up smaller objects, it may cause the end of the robot arm to be too far or too close to the object. Therefore, in order to improve the accuracy of picking up, you can try experimenting with a larger foam block. (4*4*4CM foam cubes are recommended)



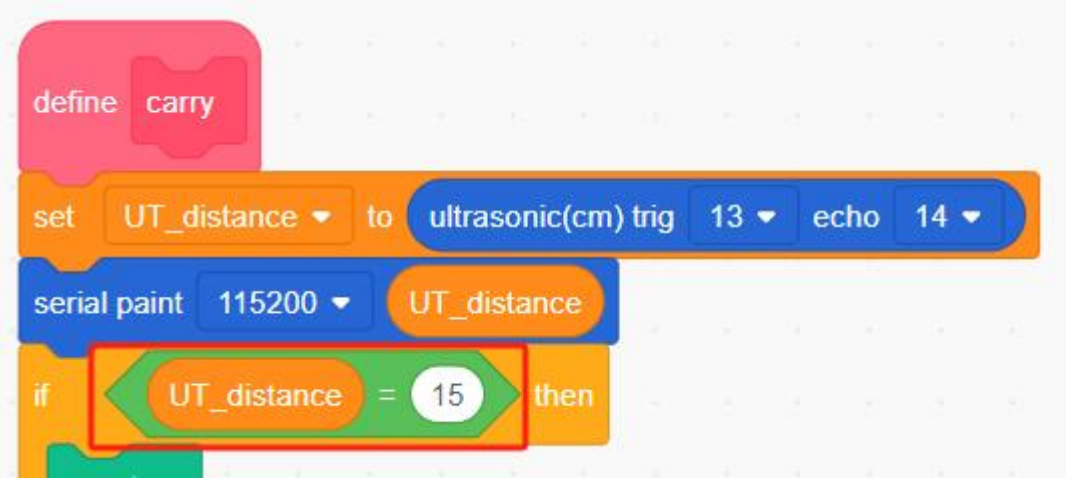
Ⅱ. Program Adjustment

If you need to replace three small blocks with one larger bubble block, you need to set the variable count in the program that sets the number of blocks to 1.

If the end of the robot arm is too high when clamping an object block, you can adjust the angle value of the variable shoulderAngle. The larger the value, the lower the height of the end. The smaller the value, the higher the height of the end.

If the robot arm grasps the object at the right end height, but the distance between the end of the robot arm and the object is biased (too close or too far), you can change the distance range in the carry function to adjust.When the end of the robot arm does not

reach the position of the target block, the distance range can be reduced, such as "13"; when the end of the robot arm crosses the position of the target block, the distance range can be increased, such as "16".



Lesson 3 Web Control Of Robot Arm Car

With the continuous development of wireless communication technology and the Internet of Things (IoT), remote control device technology has found extensive applications across many fields. It allows users to precisely control terminal devices from a distance. There are many types of wireless communication technologies, and this tutorial will focus on how to use WiFi communication technology to achieve remote control of a robotic arm vehicle.

WiFi communication technology is a type of wireless local area network (WLAN) technology that allows electronic devices such as smartphones, tablets, and laptops to connect to the internet or local area network wirelessly. WiFi technology connects devices to the same network via wireless routers or access points (APs), enabling data to be exchanged between devices.

Web-based control of devices is one of the primary applications of WiFi communication technology, with widespread use in smart homes and smart industries. Web-based control involves connecting devices and control terminals via the internet. Interaction between the device and the controller can be achieved using a simple HTTP protocol. When a device is connected to a controller, the controller provides a straightforward web interface, allowing users to access and control the device through a web browser.

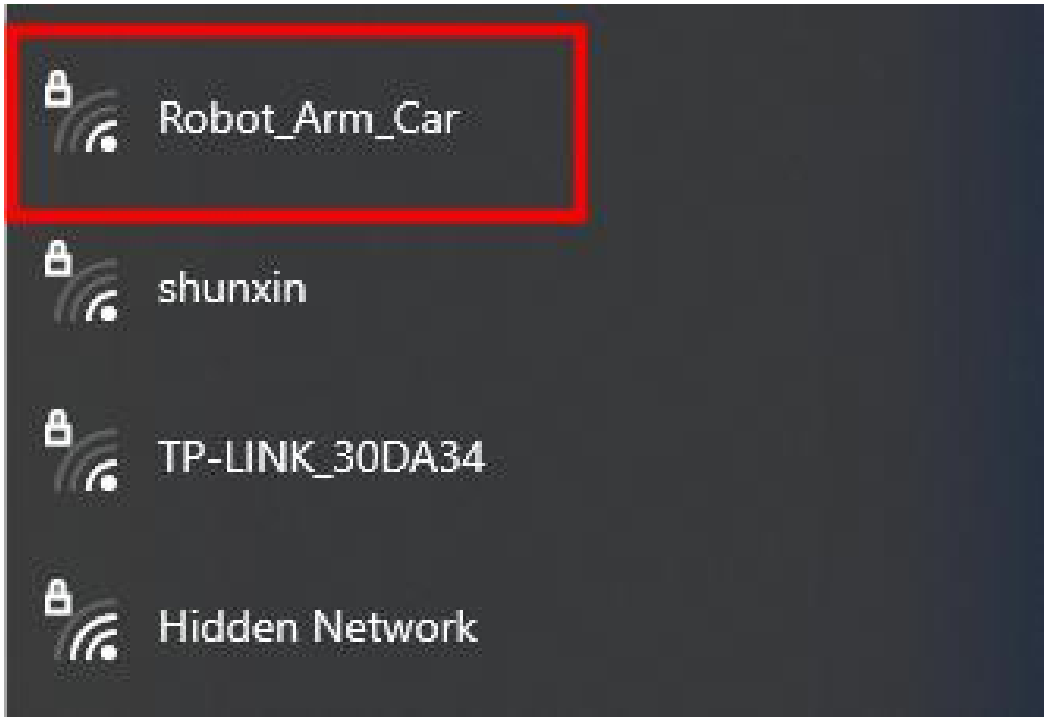
Next, we will use a web page to remotely control the robotic arm vehicle.

Ⅰ. Web Control Program

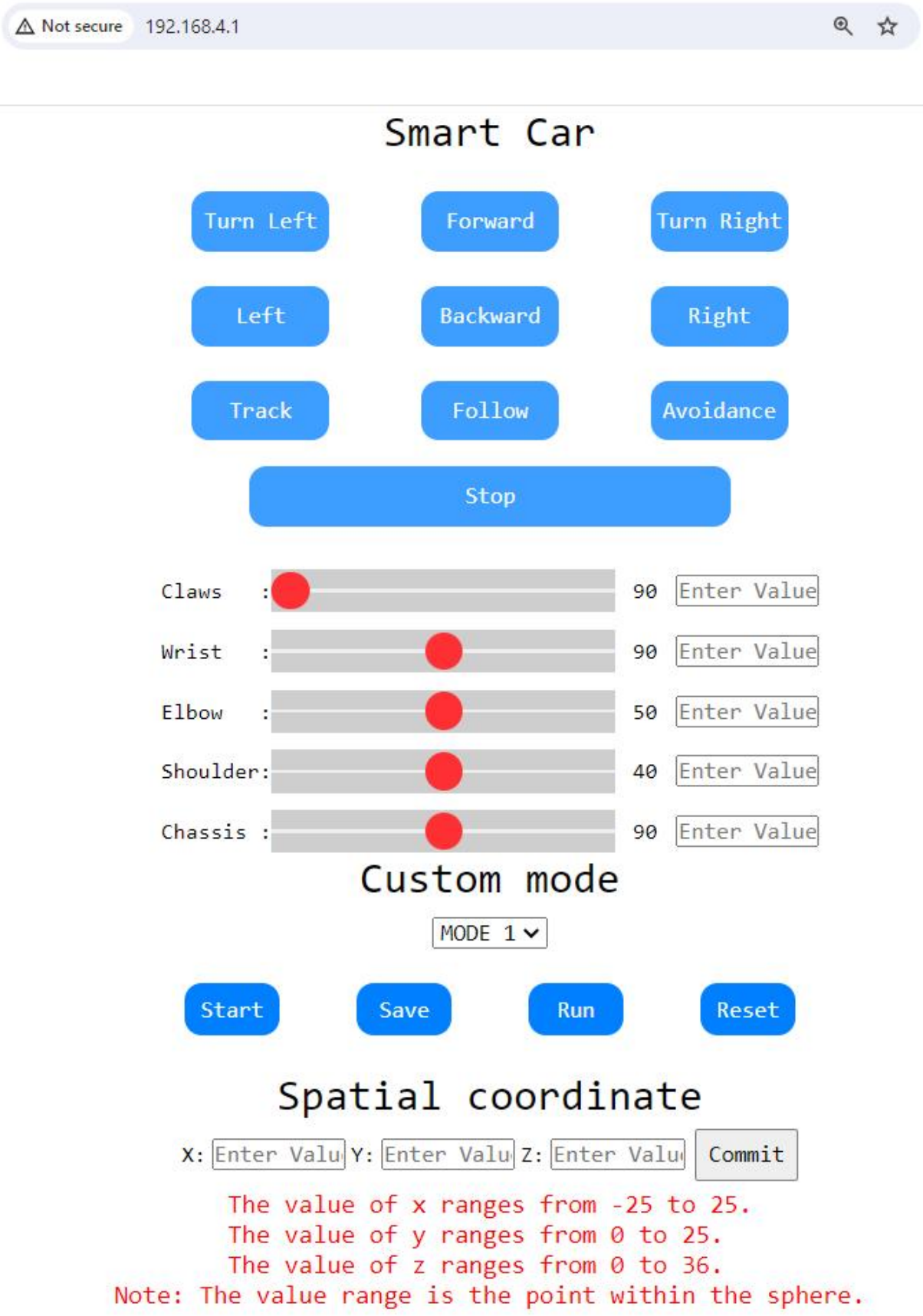
Open “Web_Controlled_Robot_Arm_Car.sb3” in “English\ACECode(Blockly Code)\3.Program\lesson3”,connect the ESP32 controller board to the computer using a USB cable. Ensure the correct controller board model and port number are selected. Upload the code to the ESP32 controller board. The controller board should be powered by the battery pack, with the battery pack switch set to the "ON" position.

Ⅱ. Login Page

After the upload is successful, then use the computer or mobile phone wireless network to scan the WIFI, connect to the WIFI hotspot named "Robot_Arm_Car", the password is 12345678, as shown below.



After the connection is successful, enter "192.168.4.1" in the address bar of the browser. The page interface is as follows:


Not secure
192.168.4.1
Smart Car
Turn Left
Forward
Turn Right
Left
Backward
Right
Track
Follow
Avoidance
Stop
Claws : 90 Enter Value
Wrist : 90 Enter Value
Elbow : 50 Enter Value
Shoulder: 40 Enter Value
Chassis : 90 Enter Value
Custom mode
MODE 1
Start
Save
Run
Reset
Spatial coordinate
X: Enter Valu Y: Enter Valu Z: Enter Valu Commit
The value of x ranges from -25 to 25.
The value of y ranges from 0 to 25.
The value of z ranges from 0 to 36.
Note: The value range is the point within the sphere.

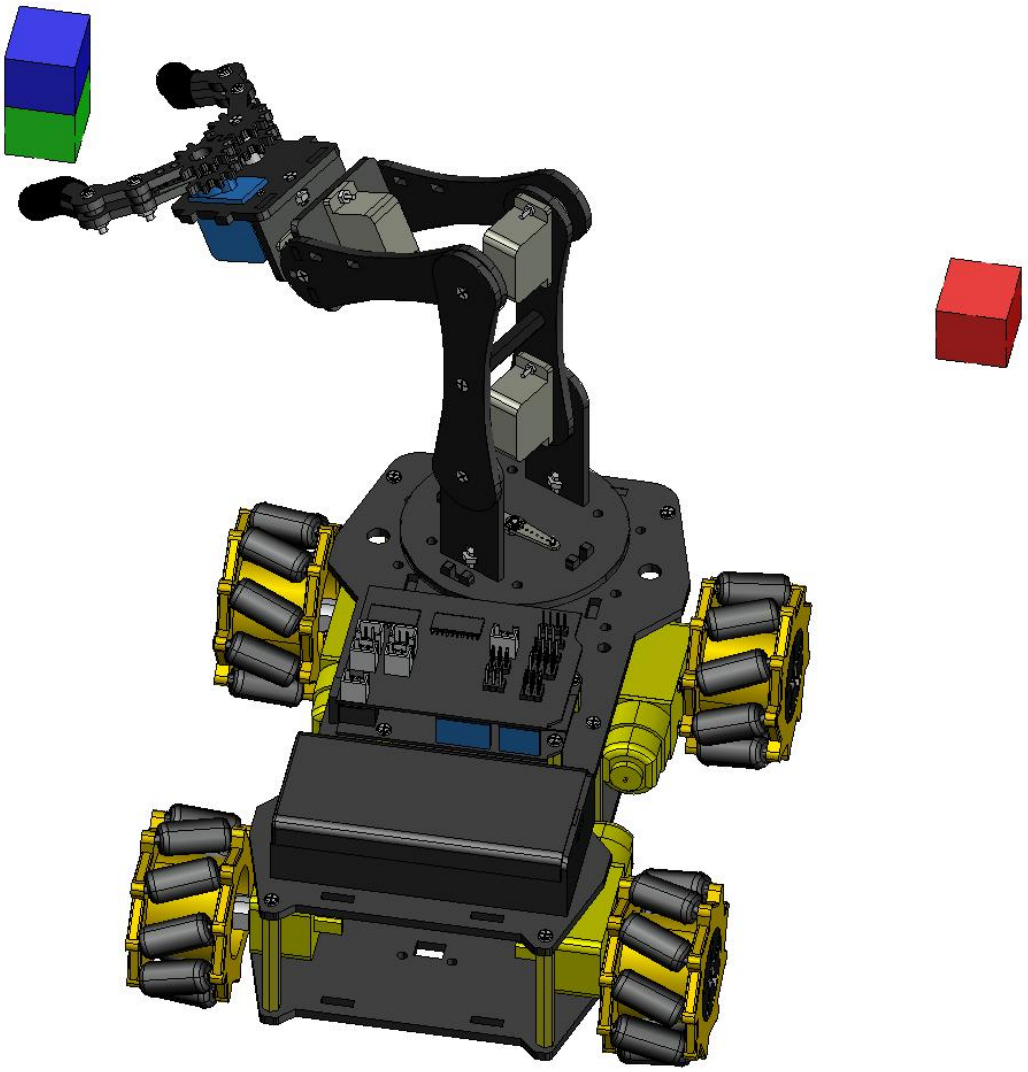
NO.	Web function	Function description
1	Button control	Through the button on the web page to control the movement of the robot arm car in the direction of front, back, left and right.
2	Slider control	The motion of the corresponding joint of the manipulator is controlled by moving the slider or entering an Angle in the input box. Attention: Move the slider slowly, the faster the slide, the faster the robot arm joint moves.
3	Action saving	A total of 6 groups of robot arm actions (Mode 1~6) can be saved, and each group can save 20 different actions. The specific operation process is as follows: ① Click Start, the button will become End, and then the Save action, according to your action path, click Save step by step, pay attention to the start position and the end position to click Save; ② Click End to finish the action and save; ③ Click Run to display the action; ④ Click Reset to reset the action group.
4	Spatial orientation	Enter the spatial coordinate values of x, y, z, and click Confirm. The robot arm will move according to the specified spatial coordinate point. Attention: x, y, z input box has the corresponding value range description, if out of the range value, you need to re-enter.

Ⅲ. Expand The Task

According to the way of web control of the robot arm car, next we control the robot arm car in the web to achieve the function of picking up the object block.

Task Description:

Place three blocks on the ground at place A, and use the web control robot arm car to pick up the blocks and move them to place B, the position can be defined by itself. Note: To avoid damage to the servo, please do not hold the block for too long, because the servo will be hot.



Lesson 4 APP Control Of Robot Arm Car

In the previous lesson, we have learned to use web pages to control the robot arm car. In order to control the robot arm car more conveniently, we choose to use the mobile phone APP as the client to realize the control of the robot arm car through the mobile phone APP. Next, we will understand how to control the work of the robot arm car through the mobile phone APP.

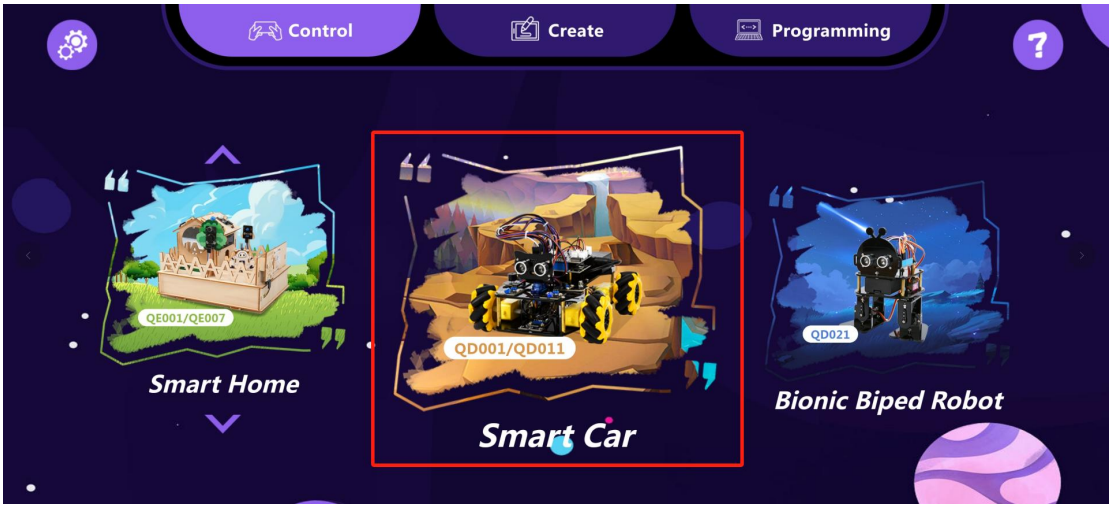
Ⅰ. APP Download

(1)If you are using an iOS device, search for the keyword "ACEBOTT" in the App Store and download it. If you are using an Android device, search for the keyword "ACEBOTT" in the Google Play Store and download it. The icon is shown as below.

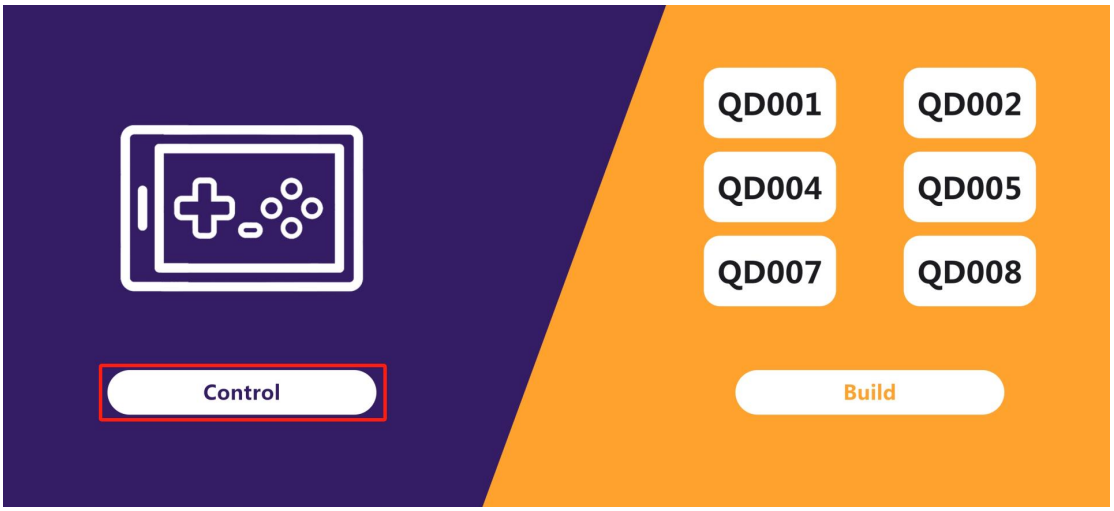
Note: 1. This tutorial is applicable to ACEBOTT APP version 2.0 and above. You can click the settings button in the upper left corner of the APP to view the software version number. Please make sure that the software version you are using meets the requirements; 2. If you need to update the ACEBOTT software version, you can refer to the method prompted in this tutorial to download the latest APP version.

(2)Open the app to enter the splash screen.

(3)Enter the selection screen and choose the”Smart Car” .

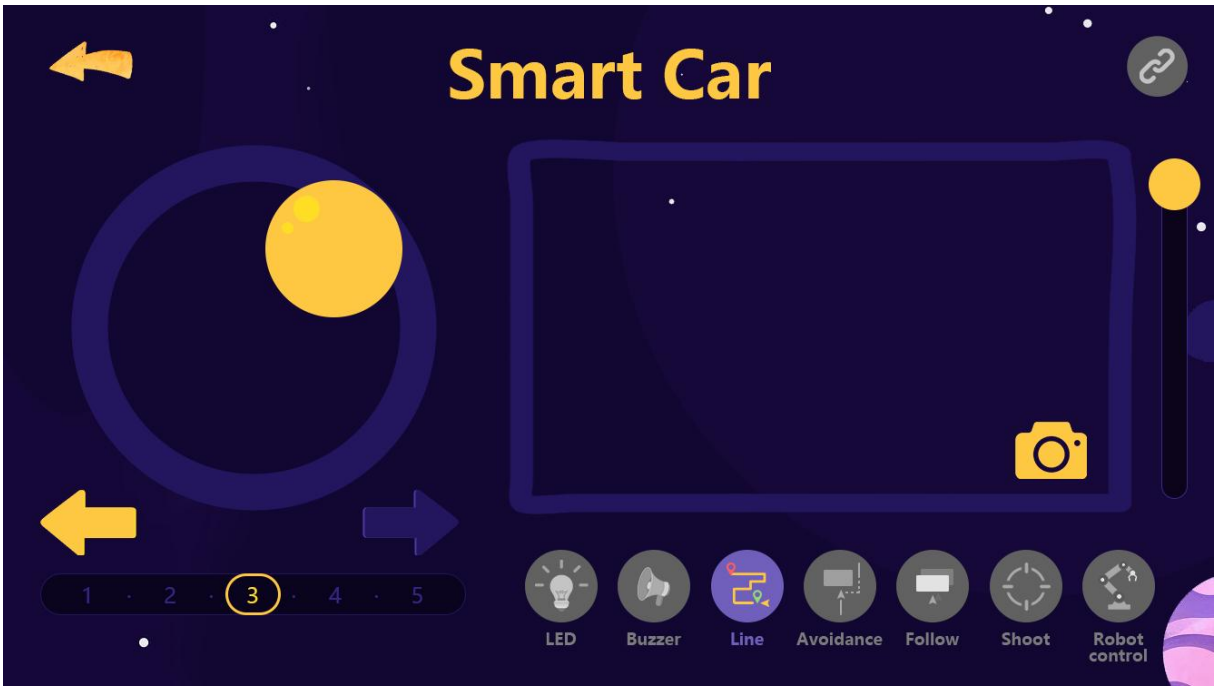


After clicking to enter, select "Control" to access the car control page.



Note: You can click the build button on the right to view the assembly video of this project.

(4) Enter the control interface of the robotic arm car (it cannot be directly controlled now, and the program needs to be burned).



Ⅱ. APP Control The Robot Arm Car

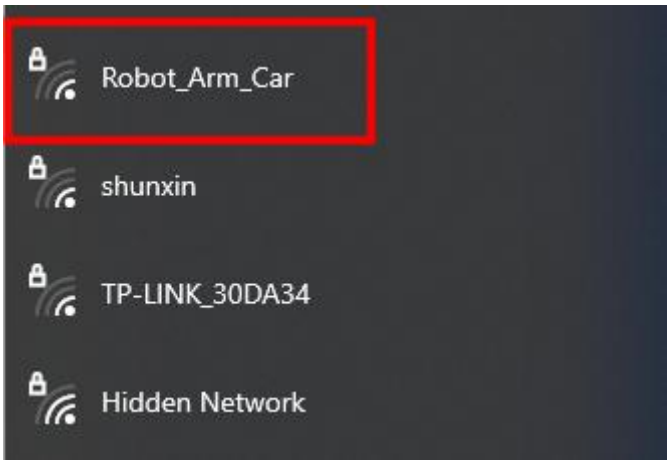
1.Upload the ACEcode program for APP control of the robot arm car.

Before using the APP to control the robot arm car, you need to upload the ACECode program that enables communication between the robot arm car and the APP to the robot arm car.

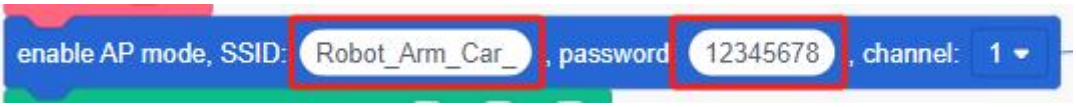
Open ”APP_Controlled_Robot_Arm_Car.sb3” in “English\ACECode(Blockly Code)\3.Program\lesson4”,connect the ESP32 controller board to the computer using a USB cable. Ensure the correct controller board model and port number are selected. Upload the code to the ESP32 controller board. The controller board should be powered by the battery pack, with the battery pack switch set to the "ON" position.

2.Connect to the Robot Arm Car's WiFi

Scan for WiFi networks on your computer or mobile phone and connect to the WiFi hotspot named "Robot_Arm_Car" with the password 12345678, as shown in the image below.

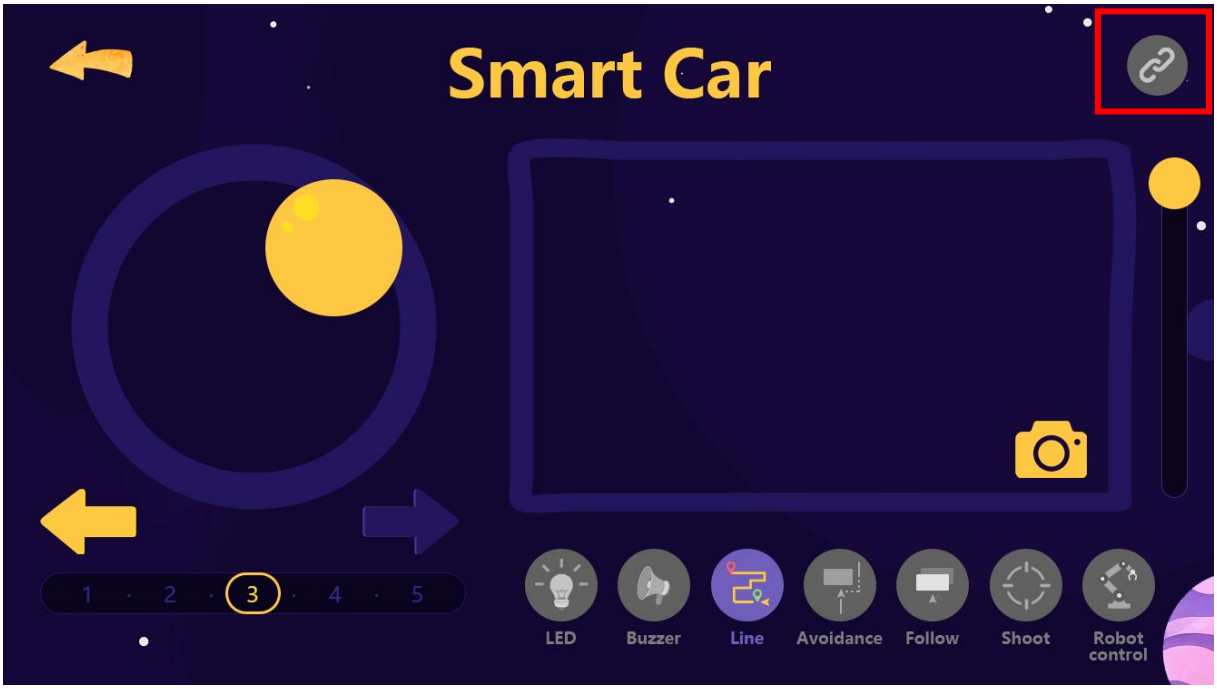


Attention: The hotspot name and password are predefined in the program, but users can customize them. When we have multiple robot arm cars, we can distinguish each one by using different WiFi names.

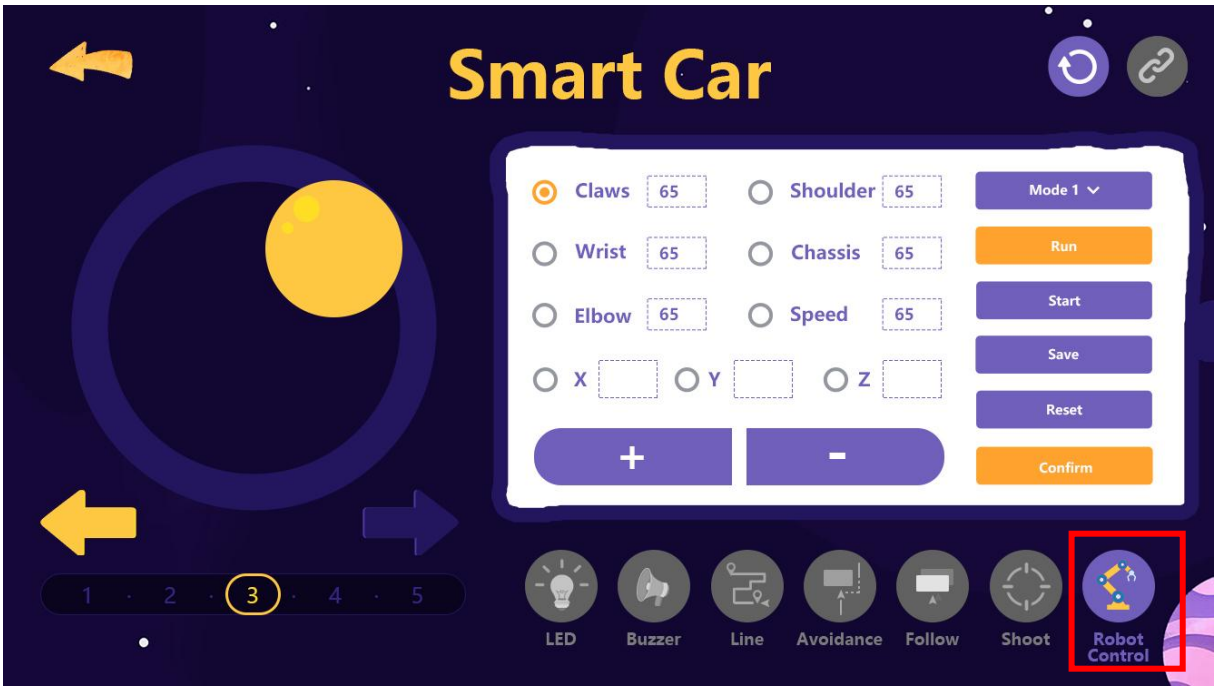


3.Using APP controls

After connecting the WiFi, click the connection icon in the upper right corner of the APP to complete the connection.



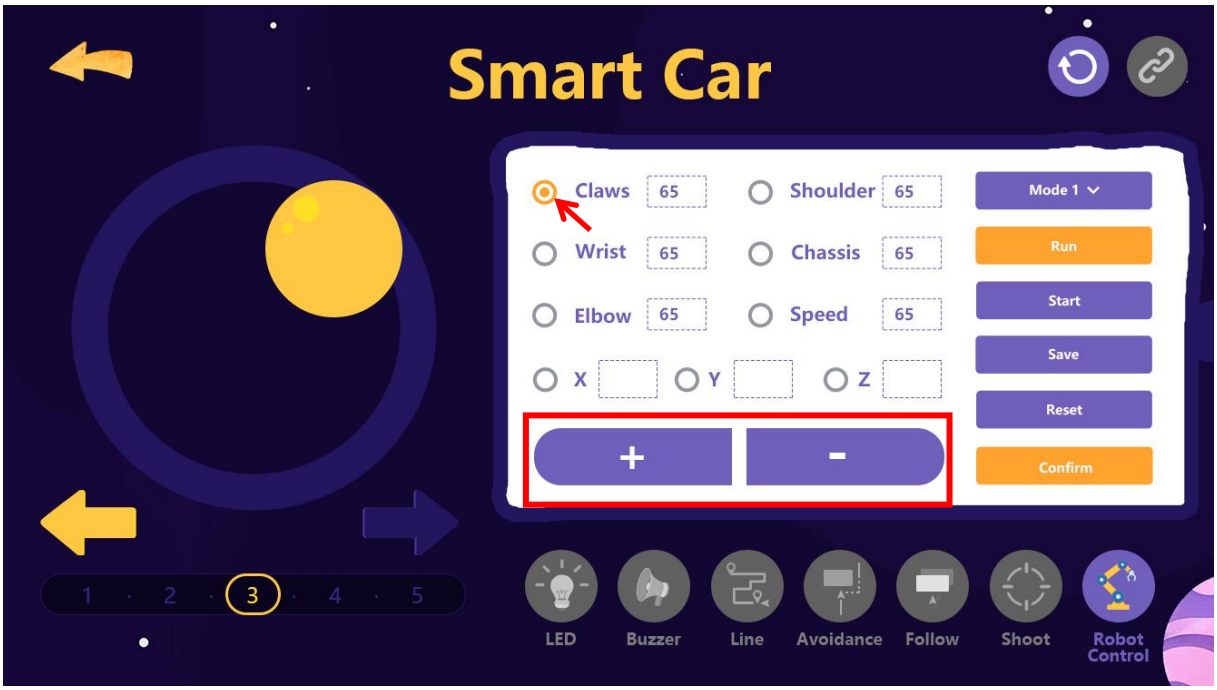
If you need to control the robot arm, click the robot arm icon in the lower right corner.



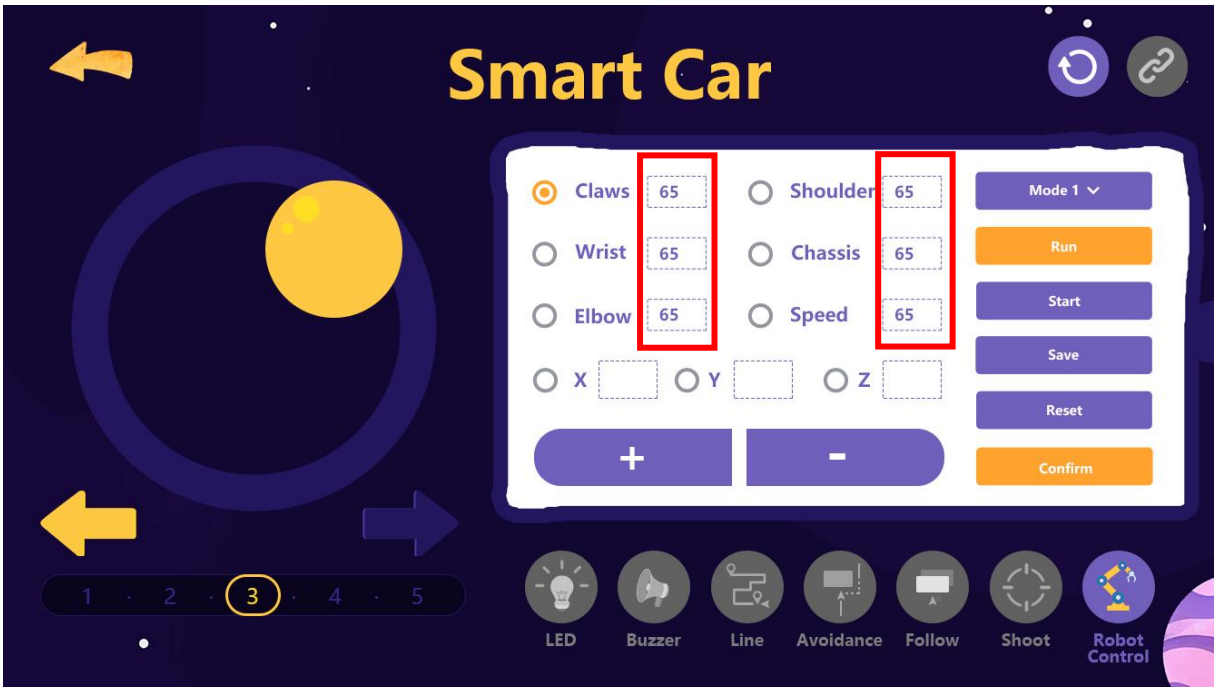
After completing the above operations, return to the interface shown below again, and then you can realize the control of the robot arm car. The main control actions are: button control, input box control, custom mode (start, end, save, run, reset), spatial positioning function and position recovery function.

Introduction of robot arm APP function:

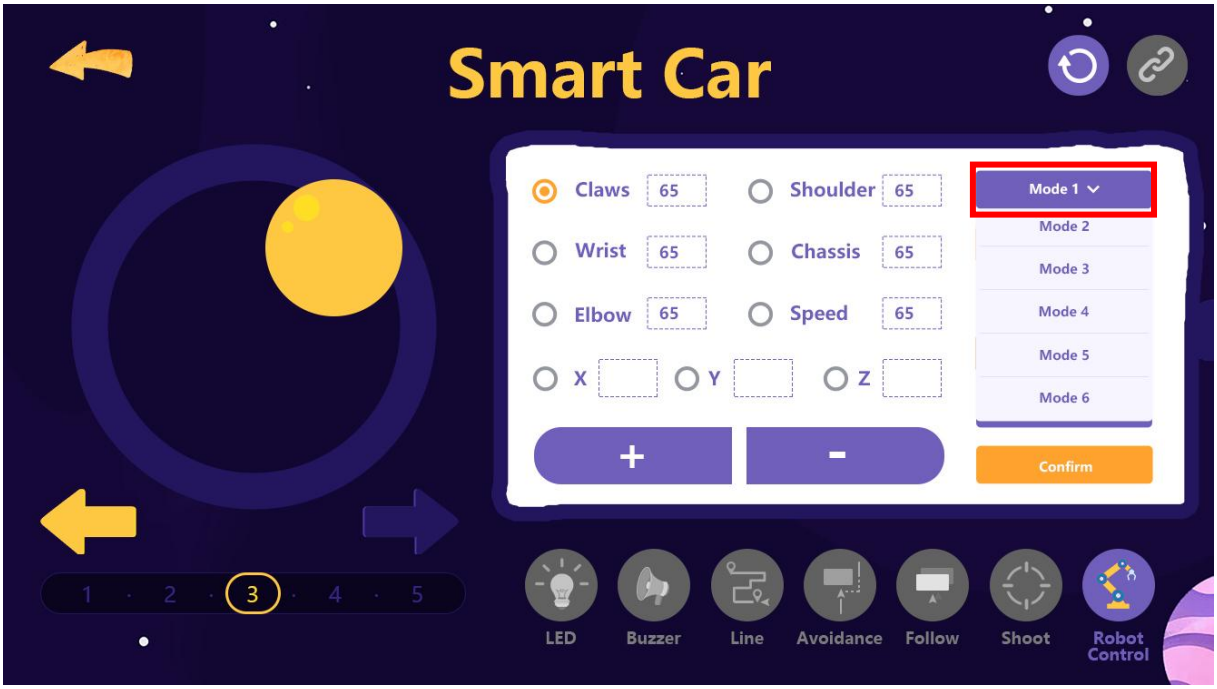
(1)Button control: After selecting different parts of the robot arm, click the "+" and "-" buttons to control the attitude change of the robot arm.The value in speed is the value that is increased or decreased each time after clicking the "+" and "-" buttons.



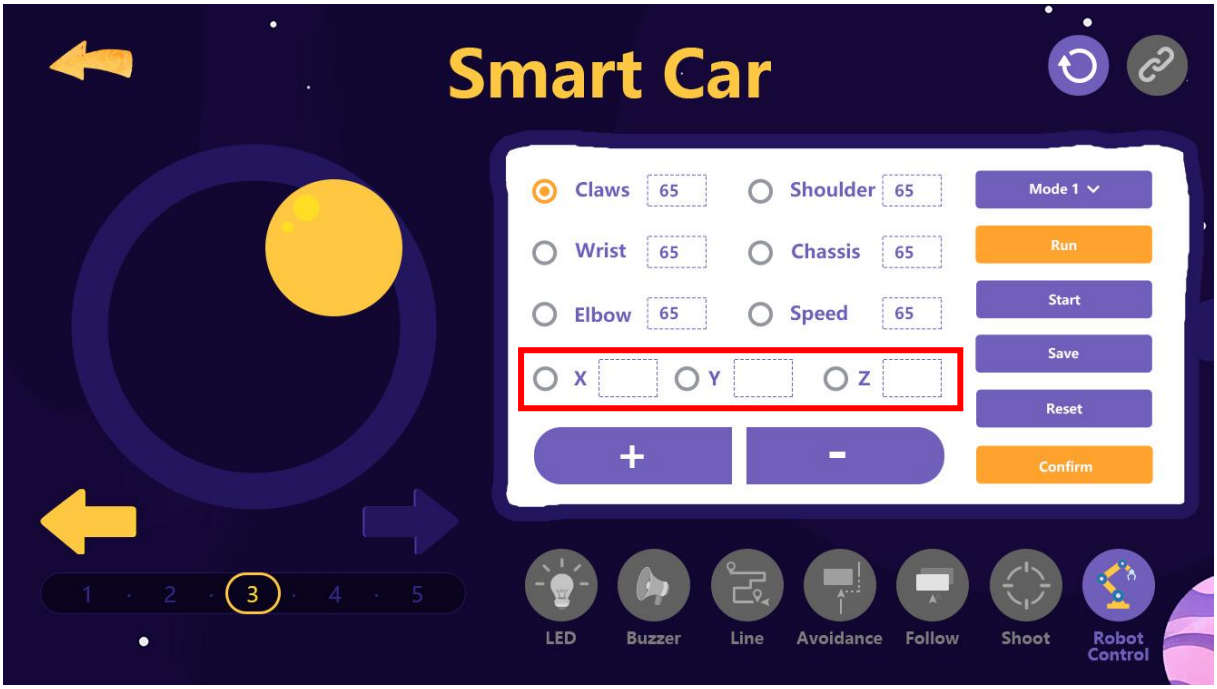
(2)Input box control: There is an input box on the right side of the robot arm, you can input the corresponding steering gear angle to control the attitude change of the robot arm.



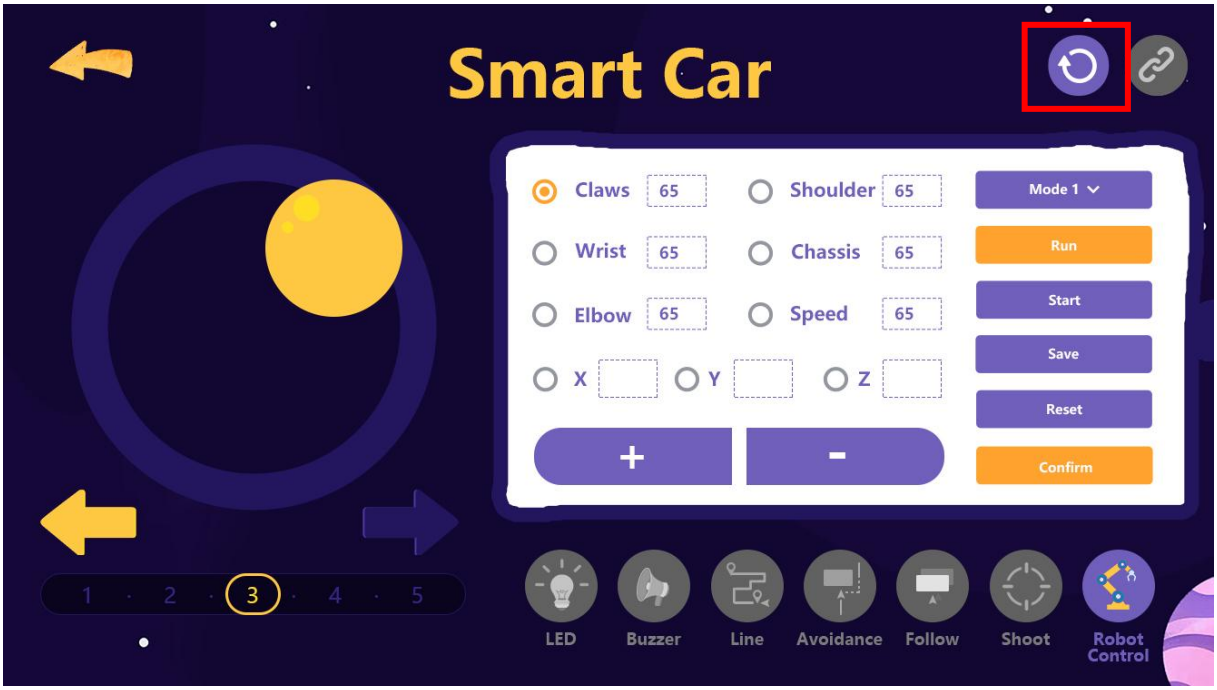
(3)Action saving: Click on "Mode1" to choose from 6 modes (Mode 1~6). Under each mode, you can save up to 20 different action sets. The process is similar to web control for saving actions.



(4)Spatial positioning: Input the spatial coordinate values of x, y and z, and the end of the robot arm will move according to the specified spatial coordinate points. Attention: Due to the influence of the structure of the robot arm, the range of motion of the robot arm is limited. When entering coordinates, if the entered value exceeds the range of motion of the robot arm, it can be re-entered.



(5)Position initialization: Click the refresh icon on the upper right, and the position of the robot arm will be restored to the initial posture.



Follow Us

Scan the QR codes to Follow Us for troubleshooting & the latest news.
We have a very large community that is very helpful for troubleshooting and we also have a support team at the ready to answer any questions.

ACEBOTT FB Group QR Code

YouTube QR Code