

Education
Solution Series

Smart Factory Education Kit Level 1



— Blockly Programming —

Contents

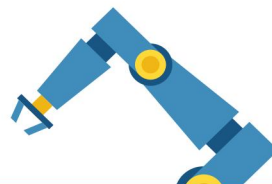
Chapter 1 Smart Factory	1
Section 1 Overview of Smart Factories	2
Section 2 Brain of Smart Factory	12
Chapter 2 Workers of the Smart Factory	29
Section 1 Robotic Arm "Worker"	30
Section 2 Joint Motion of Robotic Arm	38
Section 3 Speed Control of Robotic Arm	52
Section 4 Smart Handling	64
Section 5 Robotic Arm Coordinate Control	74
Section 6 Robotic Arm Palletizing	85
Section 7 Remote Control of Robotic Arm	93
Chapter 3 Smart Assembly Line	106
Section 1 Conveyor	107
Section 2 Speed Control of Conveyor	117
Section 3 Smart Conveyor	124
Section 4 Smart Sorting	132
Chapter 4 Smart Warehouse	140
Section 1 Smart Storage	141
Section 2 Smart Outbound Logistics	153
Section 3 Smart Warehousing Integrated System	163



Chapter 1 Smart Factory

As technology advances rapidly, society's production methods are undergoing a profound transformation. The operation of factories is gradually shifting from traditional manual labor to a new era of intelligence. The rise of smart factories has significantly enhanced production efficiency and product quality, leading the manufacturing industry into a new chapter. Next, let's uncover the mysteries of smart factories together and explore the secrets behind them!



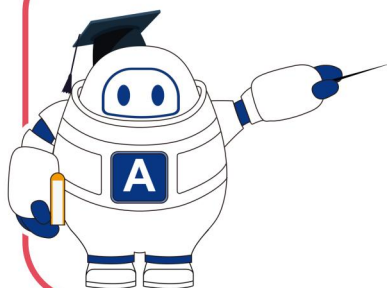


Section 1 Overview of Smart Factories



Course Objects

- Understand the functions and roles of a factory.
- Understand the concept of a smart factory.
- Understand the key technologies of smart factories.
- Understand the development history of smart factories.
- Recognize the impact of smart factories on society.



Do you know where various everyday products are made? How are they produced?

In our daily lives, we are surrounded by a wide array of products that not only meet our everyday needs but also enhance our quality of life. Most of these various products are manufactured in factories.





Factories, as symbols of modern industrial civilization, use efficient and precise production processes to enable the large-scale and high-efficiency manufacturing of goods. This not only increases the output of products but also ensures their quality in detail. The production of goods in factories generally follows the following processes:

Product Design: Design the product's functions, appearance, and structure based on market demand.

Raw Material Procurement: Purchase all the raw materials needed for production.

Manufacturing: Process and assemble the raw materials, gradually transforming them into finished products.

Quality Inspection: Inspect the product's quality to ensure it meets the required standards.

Product Packaging: Package the products that pass the quality inspection.

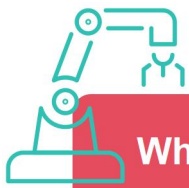
Storage and Transportation: Store the finished products in a warehouse and transport them to customers based on orders.

In traditional factories, these processes are





completed manually, often leading to issues such as low efficiency, inconsistent product quality, and high costs. As technology continues to advance, more and more factories have undergone automation and intelligent upgrades, significantly improving production efficiency and product quality. So, what exactly is a smart factory?



What is Smart Factory?

A smart factory refers to a modernized factory that uses advanced technologies and automation systems to connect various stages of the manufacturing process, achieving efficient, flexible, and intelligent production.

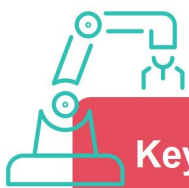
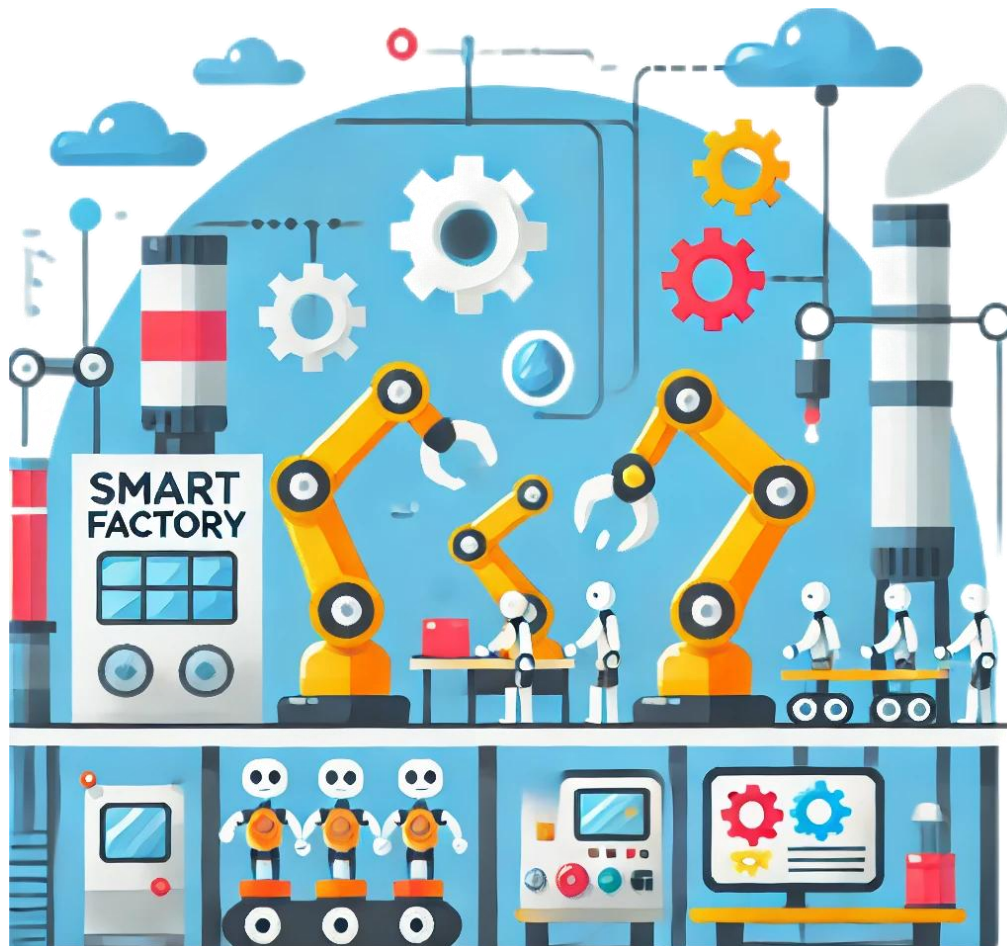
What are the characteristics of a smart factory?

- 1. Increased Production Efficiency:** In a smart factory, robots and automated equipment can operate 24/7, reducing the cost and time associated with manual operations. Robots can quickly and accurately perform repetitive tasks such as assembly and packaging.
- 2. Improved Product Quality:** Smart factories use sensors and detection equipment to monitor the production process in real time. If an issue is detected, it automatically corrects the error to ensure product quality. Additionally, big data analysis can predict and prevent quality issues from occurring.





3. Resource Savings: Through intelligent production and management, energy and material resources can be significantly reduced, lowering production costs and environmental impact, and achieving green production.



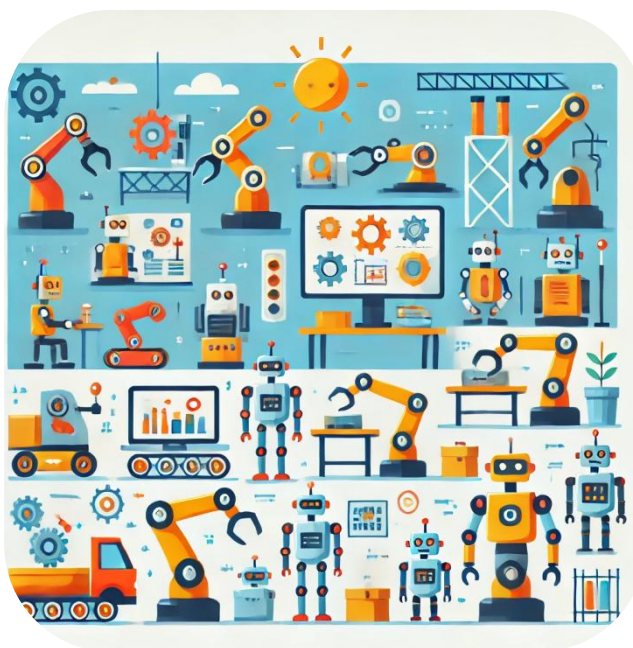
Key Technologies of Smart Factories

A smart factory is more than just simple automation; it integrates a variety of cutting-edge technologies such as robotics, the Internet of Things (IoT), big data, artificial intelligence (AI), and cloud computing to create an intelligent, interconnected production environment.





1. Robotics: Robots are crucial automation equipment in smart factories, responsible for handling complex tasks. They can autonomously complete repetitive, high-precision tasks such as welding, assembly, and material handling. Additionally they can work collaboratively with human workers, enhancing production flexibility and efficiency.

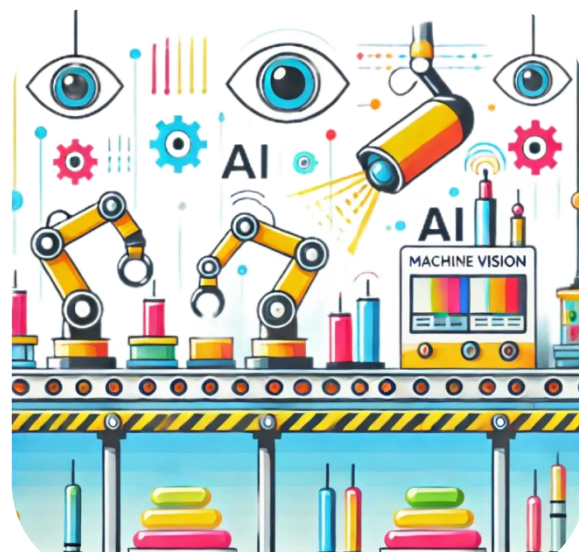


2. Internet of Things (IoT): IoT technology plays a central role in the construction of smart factories. By connecting various devices to the network, smart factories achieve interconnectivity between equipment, allowing for real-time data collection, high-speed transmission, and comprehensive data sharing. More importantly, this technology enables managers to remotely monitor equipment status from any location and exercise precise control.



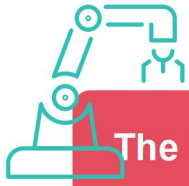


3. Artificial Intelligence (AI): AI technologies, such as image recognition and machine learning, play a critical role in smart factories. They significantly enhance production efficiency, product quality, and resource utilization. Through predictive maintenance, intelligent scheduling, quality control, supply chain optimization, energy management, and smart robotics, AI enables the automation, intelligence, and optimization of the production process. This allows factories to respond more flexibly and efficiently to changes in market demand.



4. Cloud Computing: In smart factories, cloud computing drives the intelligence and digital transformation of the production process by providing powerful computing and storage capabilities, real-time data analysis, remote monitoring, supply chain optimization, and virtual simulation. This enhances production efficiency, resource utilization, and flexibility.





The Development of Smart Factories

The development of smart factories is closely linked to the advancement of science and technology. Its history can be traced back to various stages of the Industrial Revolution, with each stage injecting new energy and possibilities into its evolution.

The First Industrial Revolution (Late 18th Century - Mid-19th Century):

The First Industrial Revolution began in the United Kingdom. With the invention and application of the steam engine, factory production processes shifted from manual labor to mechanized production, greatly increasing production efficiency.



The Second Industrial Revolution (Late 19th Century - Early 20th Century):



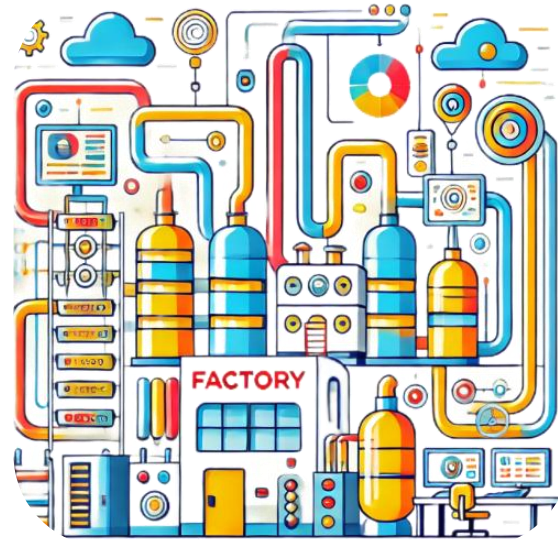
The discovery of electricity and the subsequent invention of the generator and electric motor marked the official entry of the industrial sector into the electrical age, greatly driving leaps in productivity and production efficiency.





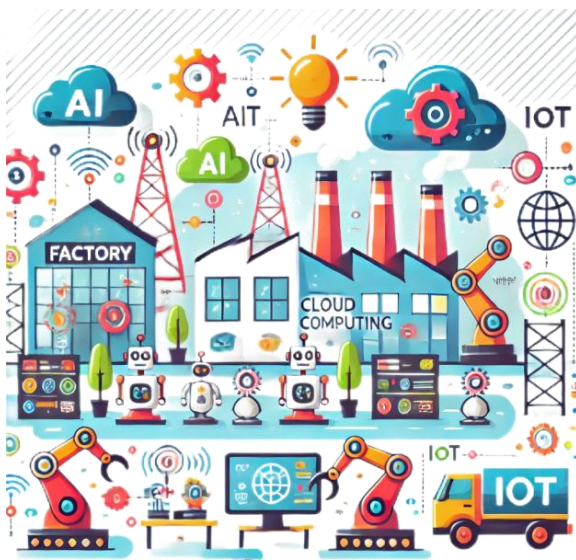
The Third Industrial Revolution (Mid-20th Century - Late 20th Century):

The Third Industrial Revolution was centered around the rapid development of electronic computers, integrated circuits, and information technology. This revolutionary progress not only marked the beginning of the automation era but also ushered in the wave of the



information age. During this stage, factories achieved a high level of automated production, and the initial forms of intelligent technologies began to emerge, signaling a significant transformation in future production methods.

The Fourth Industrial Revolution (21st Century to Present):

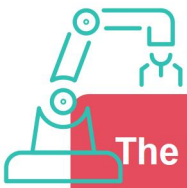


The Fourth Industrial Revolution, driven by artificial intelligence, big data, the Internet of Things, and cloud computing, is pushing the manufacturing industry towards greater intelligence, digitalization, and connectivity, enhancing production efficiency and flexibility.





The development of smart factories reflects the history of humanity's continuous exploration and innovation in manufacturing. From the transitions of mechanization and electrification to digitalization and intelligence, smart factories have evolved, bringing about profound changes in production methods and efficiency, and becoming a new milestone in the manufacturing industry.



The Impact of Smart Factories on Humanity

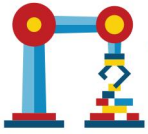
Smart factories have played a significant role in improving production efficiency and driving economic growth, while also enhancing working conditions, fostering technological innovation, achieving energy savings and emissions reduction, and improving the quality of life. However, we cannot overlook some of the negative impacts brought by smart factories. As robots gradually replace human labor, some jobs may be reduced, creating employment pressure. Additionally, due to the heavy reliance of smart factories on networks and data, there is an increased risk of data security breaches and privacy leaks, necessitating stronger management and protection.



Brainstorm

What do you think will be the impacts of smart factories on human life?





Do It.

What does your ideal smart factory look like? Please describe your ideal smart factory below, listing the technologies it uses and how these technologies help the factory achieve efficient production. Additionally, please draw a simple diagram of your smart factory below, labeling the key components.



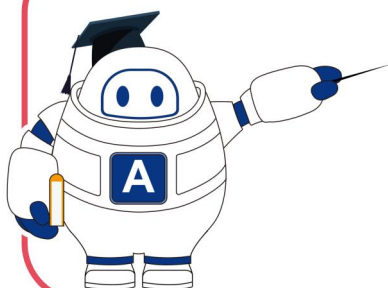


Section 2 Brain of Smart Factory



Course Objects

- Understand the functions of the ESP32 controller board.
- Learn how to use the ESP32 controller board.
- Learn how to install the ACECode.
- Write your first ACECode program.
- Learn how to upload a program to the ESP32.



Do you know why the Smart Factory is smart?

A smart factory is considered intelligent because it possesses a "brain" comparable to human intelligence—the control system. This system plays a crucial role in the operation of a smart factory, not only receiving and processing various data but also accurately guiding equipment to

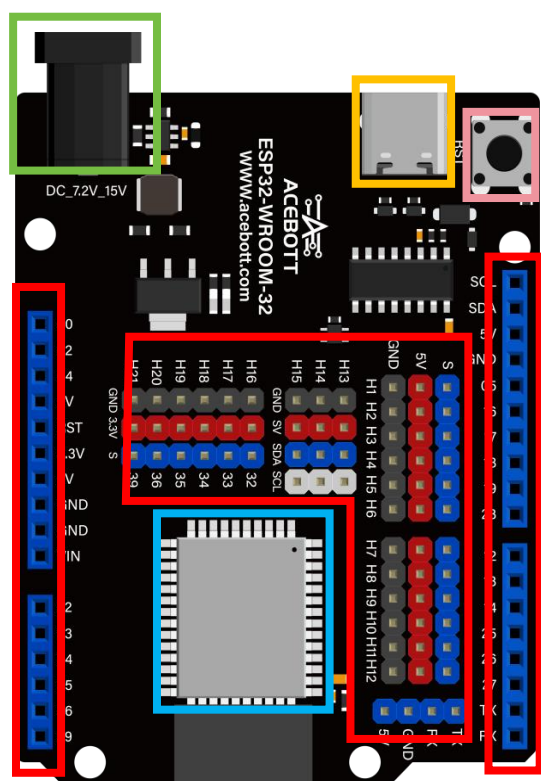




perform automated tasks. At the core of the control system are controllers, which are typically hardware devices with powerful data processing capabilities, such as microcontrollers and computers. Next, we will delve into one of the most popular controllers, the ESP32, and explore the secrets behind it.



ESP32 is a low-power, highly integrated Wi-Fi and Bluetooth dual-mode microcontroller developed by Espressif Systems. The ESP32 controller board is very popular due to its powerful features and abundant communication interfaces, making it suitable for various Internet of Things (IoT) applications, embedded system development, and project implementations. Below is the schematic diagram of the ACEBOTT-ESP32 controller board:

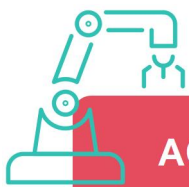


- **Power input:** It can be connected with 7.2V-15V power supply to power the controller board.
- **Type-C interface:** For program download, also can provide 5V power supply for the controller board.
- **Reset key:** Reset the system of the controller board.
- **GPIO:** Generic input and output interface for input and output data.
- **ESP32 modules:** Includes ESP32 microprocessor, Bluetooth, WiFi.



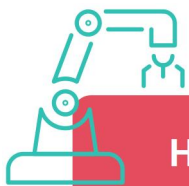


After you get the ESP32 controller board, its initial state is not able to implement any specific features. If you want it to have a specific function, you have to program it. This requires first configuring the development environment suitable for ESP32 to provide a programming platform for developers. Commonly used ESP32 development platforms include MicroPython, Arduino IDE, ESP-IDF, and so on. This course will focus on using Arduino IDE as the ESP32 development platform.



ACECode

ACECode is a graphical programming tool that allows users to program by dragging blocks without having to master complex programming languages. ACECode covers all the functions of Scratch and adds a robot control module, which allows users to design their own robot works using graphical programming, reducing the difficulty of robot programming and development.



How to install the ACECode on Windows?

Step 1: Download the ACECode installation program from the official web site. Log in to the official website of ACECode: <https://www.acebott.com/pages/software>, enter the software download interface, select the ACECode software version for Windows system, and click Download ACECode to

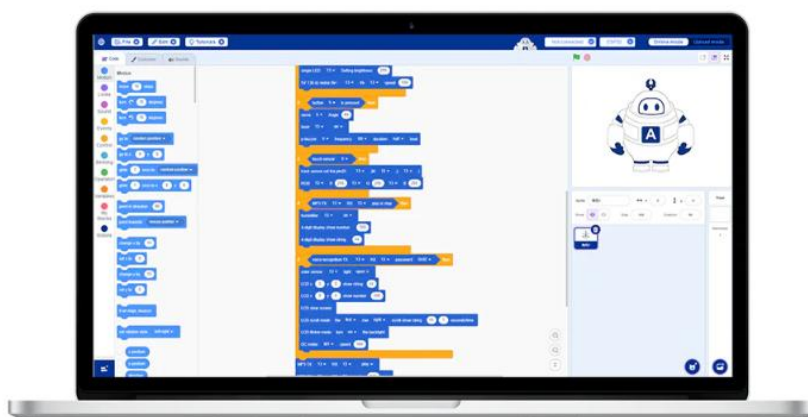




download.

[Home](#)[Products](#) ▾[Download Tutorials](#)[Software](#)[Wiki](#)[Blog](#) ▾[About Us](#)[Contact us](#)

any technical problems, feel free to contact us at support@acebott.com

[Download Windows](#)[Download Mac X64](#)[Download Mac ARM64](#)

Tips

1. If you download a compressed package, please unzip it and proceed with the subsequent installation steps;
2. This tutorial is applicable to ACECode version 2.0.2 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 use meets the requirements;
3. 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.



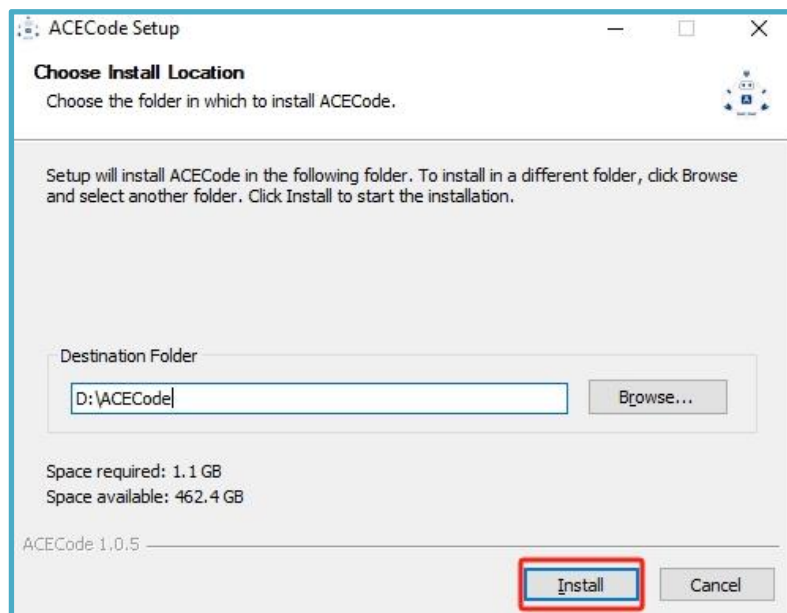


Step 2: Double-click the downloaded installer and follow the instructions to install ACECode. Here we take the installation under Windows system as an example for demonstration.

1.After the download is complete, the installation package file will appear as shown in the figure. Click to install the software.

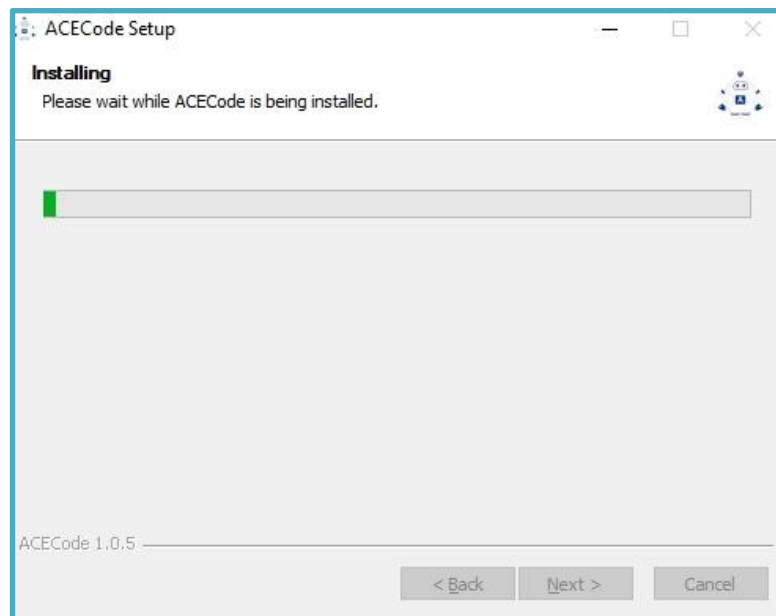


2.After clicking the software, the following interface will appear. Select "Install". You can choose the default installation path or choose the software installation route yourself.

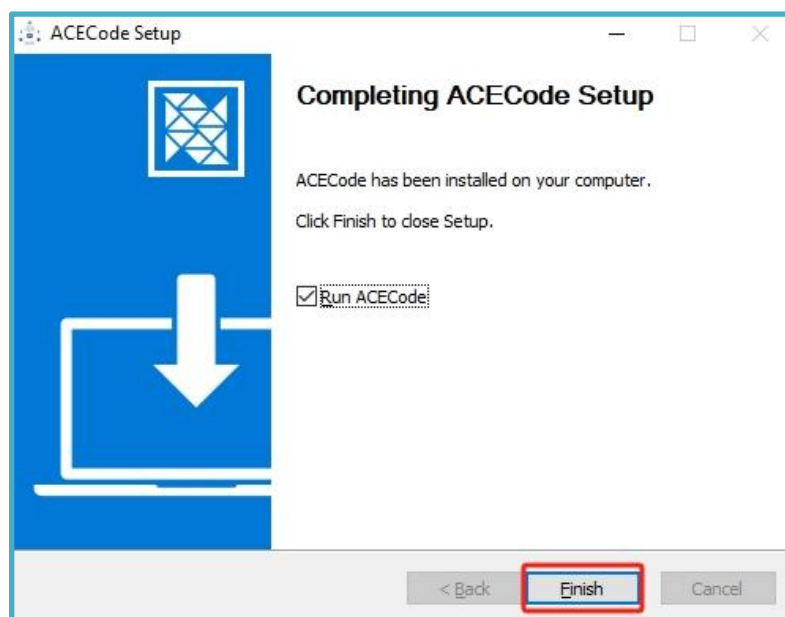


3.ACECode software is being installed.



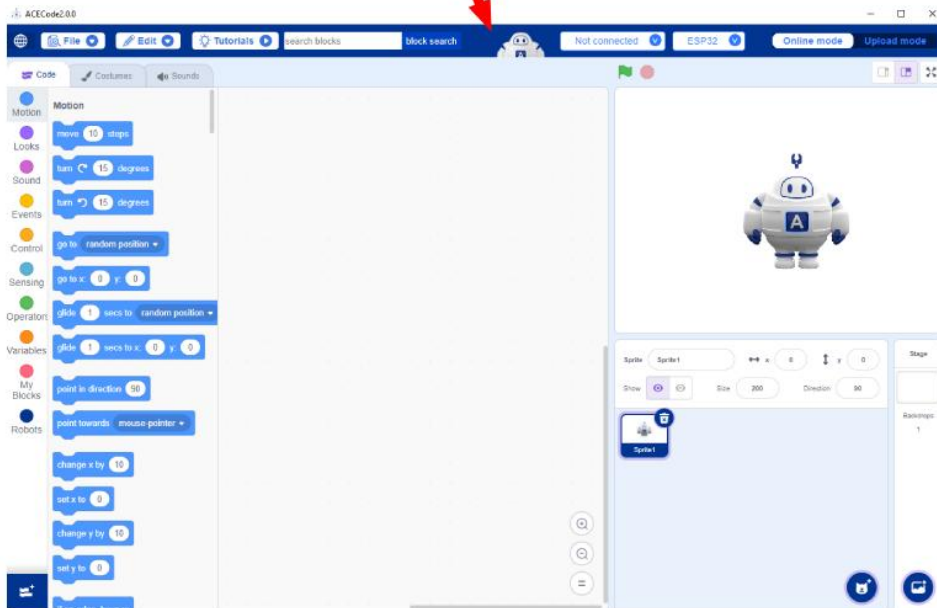


4.The installation is complete.



5.Find the shortcut of ACECode on the desktop and double-click to open ACECode.





Step 3: Install the serial driver (skip it if installed)

1. Open ACECode, click on the serial port connection button, and in the pop-up options, select "One-click to install serial driver." After clicking, it will sequentially install the serial port drivers required for the two controller boards supported by ACECode, ESP8266 and ESP32.

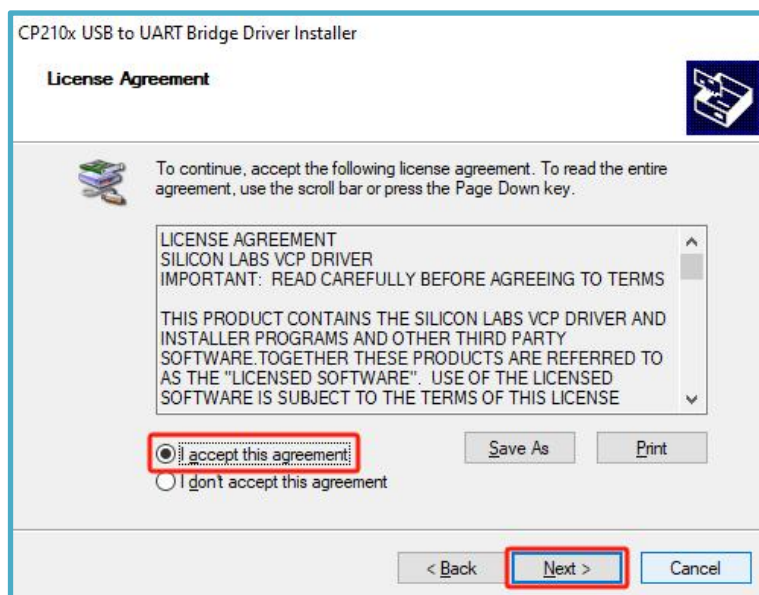


2. Click "Next" based on the prompt.



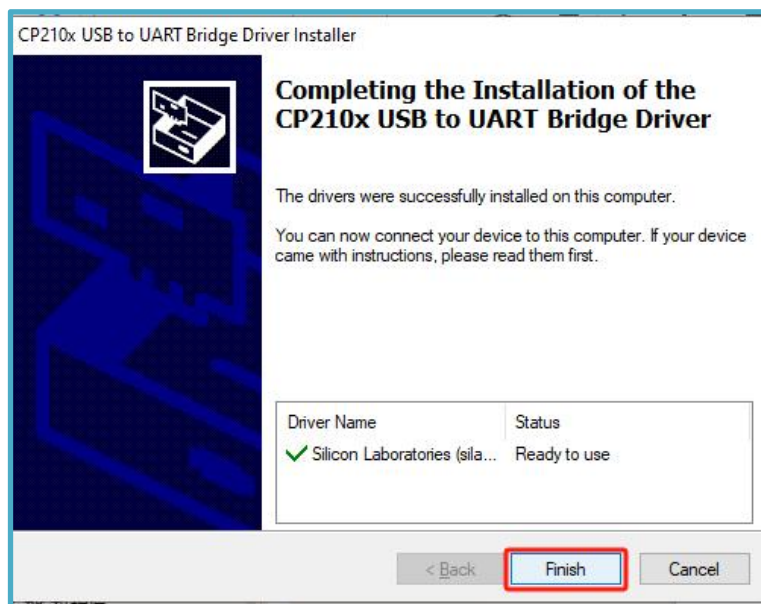


3. After clicking "I accept this agreement" click "Next" again.

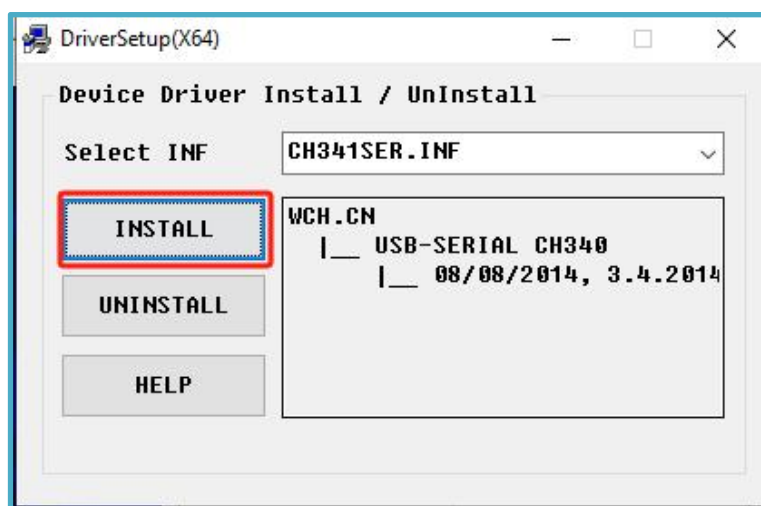


4. Click "Finish", the first driver installation is complete.

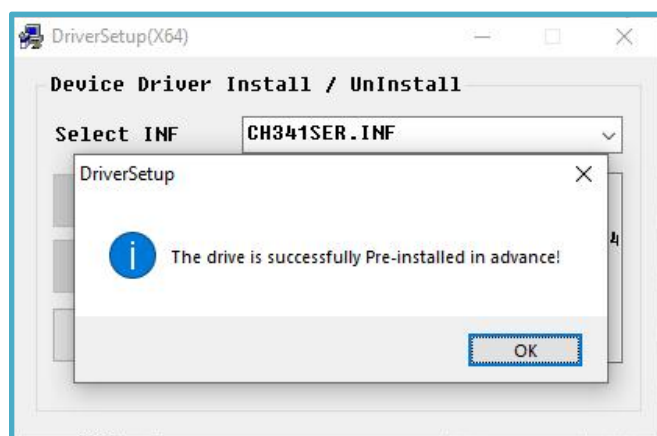




5.Next, a pop-up window for the installation of the second driver will appear; click "Install."



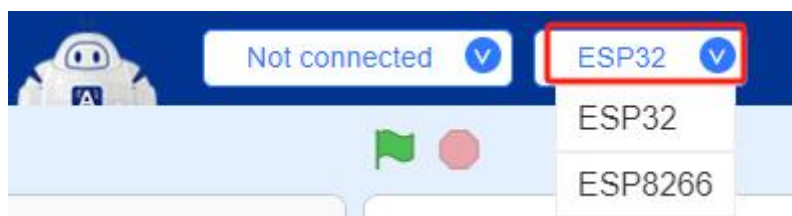
6.After the installation is completed, a message indicating that the installation was successful will be displayed.





Step 4: Connect ACECode and ESP32 controller board

1. In the ACECode controller board list, select the ESP32 controller board. The software selects ESP32 by default.



2. Find the serial communication connection button in the ACECode interface. The connection status of the serial communication will be displayed on the button. Move the mouse to the serial communication connection button position, and the information list of the serial communication port in the computer will pop up automatically; when no device is detected, it will prompt "No device were found".



3. When we connect ESP32 to the computer with a data cable, a new port "COM18" will be added to the serial communication port information list. The port number is random and will change according to the actual situation. Click the newly added port to establish a communication connection between ACECode and ESP32. After the connection is successful, the connection status display on the connection button will switch from "Not connected" to the





information of the connected port.



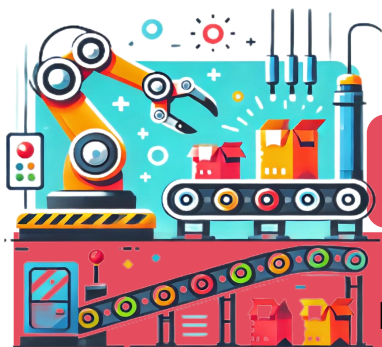
4. Currently, ACECode supports two development modes: online mode and upload mode. The online mode supports online debugging, which can debug the program in real time, which is convenient and fast; the upload mode is to upload the written program to the ESP32 (or other controller board) controller board. After the upload is successful, the program can be run on the ESP32 without the computer (the controller board needs to be powered externally).



Tips

1. You can debug the program in Online mode, and after the debugging is completed, switch to upload mode to upload the program to the ESP32 controller board.
2. The USB power supply may be insufficient, so external power supply is needed. Please connect a power source with sufficient electricity to the ESP32 and turn on the power switch.





Smart Factory Lab

Program debugging and uploading



Project Description

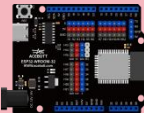


Experience the debugging and uploading of the program to make the built-in LED light on the ESP32 complete the on-off effect.



Hardware Scheme

1. Hardware List

Picture	Name	Quantity
	ESP32 Board	1
	Type-C Data Cable	1

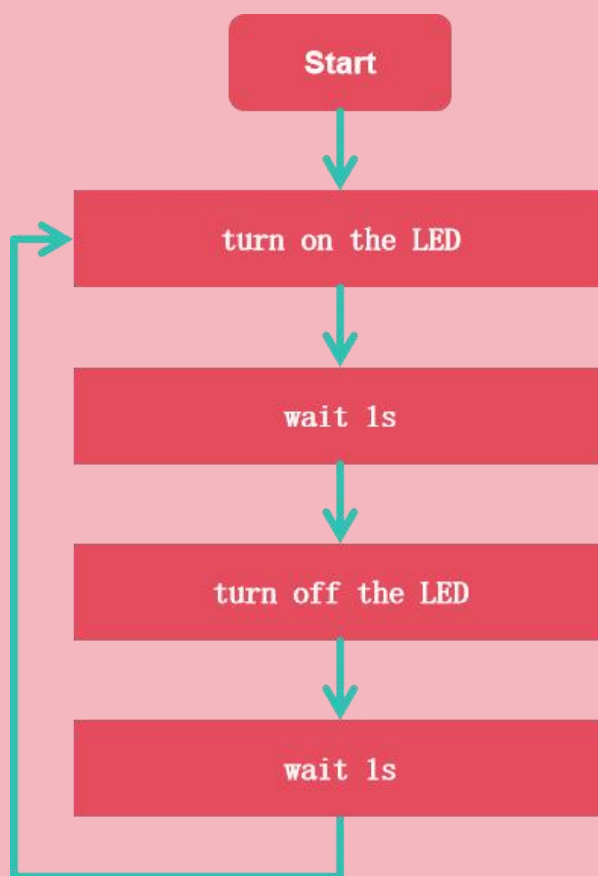




Programming



1. Program Flowchart





2. Online Mode



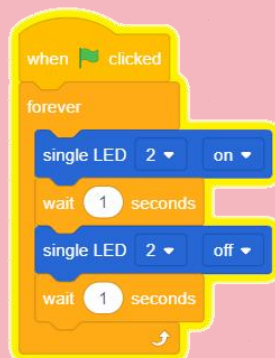
(1) Write the following code in online mode to make the built-in LED light on ESP32 flash.

Note: 1. The LED light is integrated on the ESP32 controller board and connected to I/O port 2. The effect of this instruction can make the LED light flash for 1 second on and 1 second off.

2. ACECode Software upper left corner has search function, can use this function to quickly find the required building block instructions.



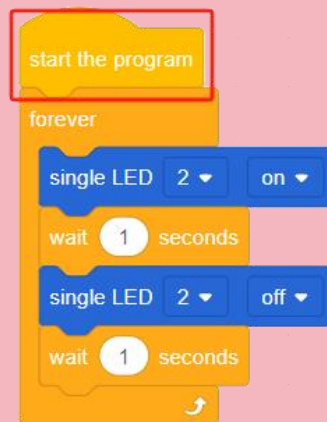
(2) After writing, debug the code online. Click the code to run it. The code will turn yellow, indicating that the code is running. The LED on the ESP32 will repeatedly light up for 1 second and then go off for 1 second. Click on the code again to stop running it.





3.Upload Mode

(1)After debugging is completed, switch to "upload mode" and change the program startup command to "start the program". In "upload mode", you need to use this command to start the program.



(2)At this point, you can see that in ACECode's "Upload Mode", the corresponding C language code and Python code will be generated synchronously to meet the different needs of users.

```
1 // generated by Block
2 #include <Arduino.h>
3 #include <ACB_SingleLED.h>
4
5 ACB_SingleLED led_2(2);
6
7 void setup() {
8 }
9
10 void loop() {
11   led_2.openLED();
12   delay(1 * 1000);
13   led_2.closeLED();
14   delay(1 * 1000);
15 }
16
```



(3) Click the "Upload" button to upload the program. When the upload progress reaches 100%, it is successful. After the upload is successful, the program can be run on the ESP32 controller board without the computer, that is, you can unplug the data cable and let the program run independently on the ESP32 (connect the ESP32 to an external power supply).



(4) If the upload fails, you can debug according to the error prompt in the lower right corner of the console. If prompted here, you can check whether ACECode and ESP32 are connected.

```

ESC[93mSPIESC[0m 2.0.0 ESC[90mC:\User
ESC[93mAdafruit GFX LibraryESC[0m 1.11.9 ESC[90mD:\aceb
ESC[93mAdafruit BusIOESC[0m 1.15.0 ESC[90mD:\aceb
ESC[93mAdafruit SSD1306ESC[0m 2.5.9 ESC[90mD:\aceb
ESC[93mmsrcESC[0m ESC[90mD:\aceb
ESC[93mAdafruit NeoPixelESC[0m 1.12.0 ESC[90mD:\aceb
ESC[93mmsrcESC[0m ESC[90mD:\aceb

ESC[92mUsed platformESC[0m ESC[92mVersionESC[0m ESC[90mPathESC[
ESC[93mesp32:esp32ESC[0m 2.0.14 ESC[90mC:\Users\GS2306JS
esptool.py v4.5.1
Serial port COM11

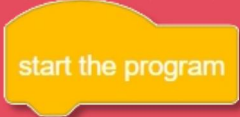
A fatal error occurred: Could not open COM11, the port
Failed uploading: uploading error: exit status 2

```





4. Instruction Parsing



start the program

:This command is located in the "Events" instruction module within the "Upload Mode." It serves as the initiation command for the entire program, and you can write the program below it.



single LED 2 on

:This instruction is located in the "Robots" instruction module. The first dropdown box allows you to select the pin to which the LED module is connected; the second dropdown box sets its on or off state. "On" means that the pin outputs a high level, while "Off" indicates a low level output.



wait 1 seconds

:This command is located in the "Control" instruction module. It allows the program to pause for a specified amount of time, during which you can enter the designated waiting time in the input box, with the unit being seconds.



forever

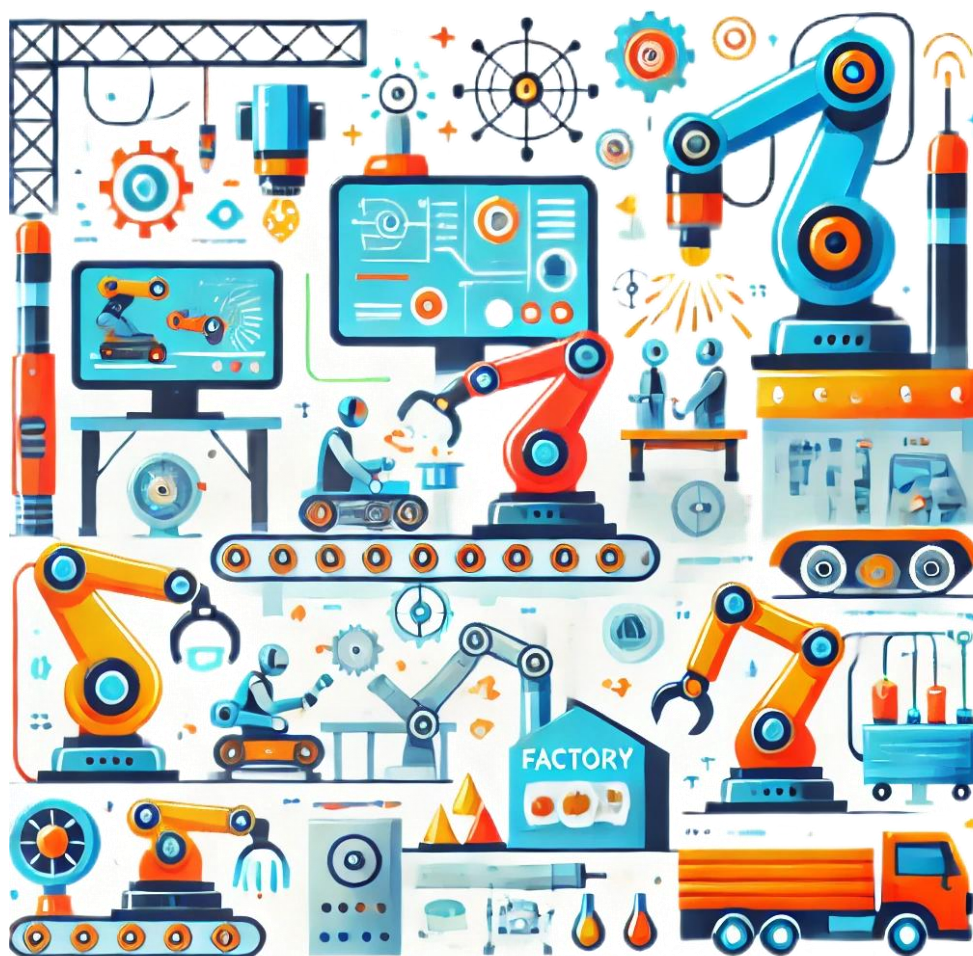
:This command is located within the "Control" instruction module, allows for the repeated execution of the instructions contained within it.





Chapter 2 Workers of the Smart Factory

In traditional factories, manual labor has long been dominant, but this approach often comes with various production issues. With the rise of smart factories, robots have gradually begun to replace human workers, significantly improving production efficiency and quality. Among the robot workforce in smart factories, robotic arms are an indispensable part. Next, we will delve into the knowledge of robotic arms and explore their applications and development in smart factories.



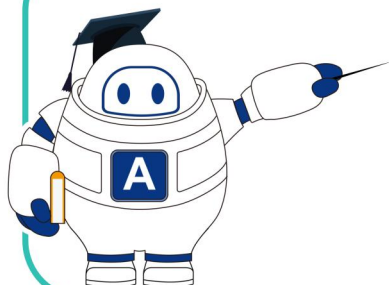


Section 1 Robotic Arm "Worker"



Course Objects

- Understand the concept of a robotic arm.
- Understand the structure of a robotic arm.
- Understand the applications of robotic arms.
- Build your own robotic arm.



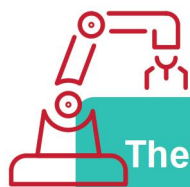
What are the workers of a smart factory like?

In the development of smart factories, the presence of human workers is gradually fading, replaced by highly efficient and precise robots and automated equipment. These machines handle numerous complex and delicate tasks, significantly enhancing overall production efficiency and product quality. Among these





modern productivity tools, robotic arms are undoubtedly the most common form. So, let's dive in and explore how robotic arms work.



The Structure of Robotic Arm

A robotic arm is an automated device that mimics the movements of a human arm and can perform various operational tasks. It typically consists of the following parts:

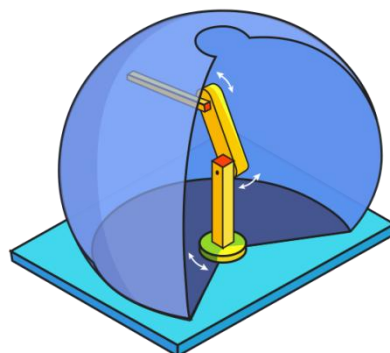
- 1. Base:** The fixed part of the robotic arm that provides stable support.
- 2. Joints:** The parts of the robotic arm that rotate or move, similar to human joints.
- 3. Links:** These connect the various joints of the robotic arm, allowing it to move flexibly.
- 4. End Effector:** The device at the end of the robotic arm that performs specific tasks, which can take forms such as a claw, suction cup, welding torch, etc.
- 5. Drive System:** Provides the power for the robotic arm's movements, usually consisting of motors or hydraulic systems.
- 6. Control System:** The "brain" of the robotic arm, controlling its movements through computer programs.





Robotic arms come in many structural types, and based on different structures and motion spaces, they can be classified into Cartesian, cylindrical, spherical, SCARA, and anthropomorphic (articulated) types. Here, we mainly introduce the anthropomorphic robotic arm.

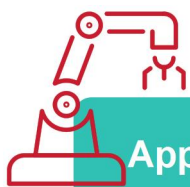
The geometric configuration of an anthropomorphic robotic arm is achieved through three rotating joints. The rotation axis of the first joint is perpendicular to the rotation axes of the other two joints, which are parallel to each other. Its motion space forms a spherical shape, as shown in the diagram on the right.



Because its structure is similar to that of a human arm, we refer to the first base joint as the **base joint**, the second joint as the **shoulder joint**, and the third joint as the **elbow joint**.

The number of independently driven joints in a robotic arm is related to its degrees of freedom (DOF). A robotic arm with three independently moving joints is called a **3-DOF robotic arm** or a **3-axis robotic arm**. Similarly, a robotic arm with four independently moving joints is called a **4-DOF robotic arm** or a **4-axis robotic arm**, and so on.





Applications of Robotic Arms

Robotic arms are widely used in modern industry and daily life. Here are some of the main application areas:

1. Manufacturing: Robotic arms can perform high-precision welding tasks, ensuring consistent welding quality and reducing errors caused by manual operations. They can also be used for assembly tasks, assembling complex components to improve production efficiency. Additionally, robotic arms can carry out painting tasks, evenly applying paint to product surfaces, ensuring high-quality appearances.



2. Healthcare: The application of robotic arms in the medical field is becoming increasingly widespread. In operating rooms, robotic-assisted surgery systems have become essential tools in modern healthcare. Robotic arms used in minimally invasive surgeries can perform precise surgical operations under a doctor's control, reducing trauma to patients. Beyond the operating room, robotic arms also





play a crucial role in rehabilitation centers. For patients with motor function impairments due to illness or injury, robotic arm-assisted rehabilitation systems can help them with their rehabilitation exercises.

3. Food Service: Robotic arms can take on various tasks in the kitchen, such as cooking and plating, significantly increasing the efficiency of food service. By using robotic arms, we can reduce the workload of chefs, allowing them to focus more on creating delicious dishes. Robotic arms can also precisely control the quantity of ingredients and cooking times, minimizing waste and improving resource utilization.



4. Agriculture: The application of robotic arms in agriculture is multifaceted.



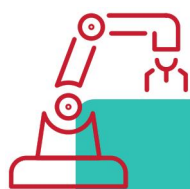
Robotic arms can be used for fruit and vegetable picking, and compared to manual picking, they can perform the task more quickly and accurately. This not only greatly increases picking efficiency but also effectively reduces damage to fruits and vegetables

during the picking process, preserving their freshness and quality. Additionally,



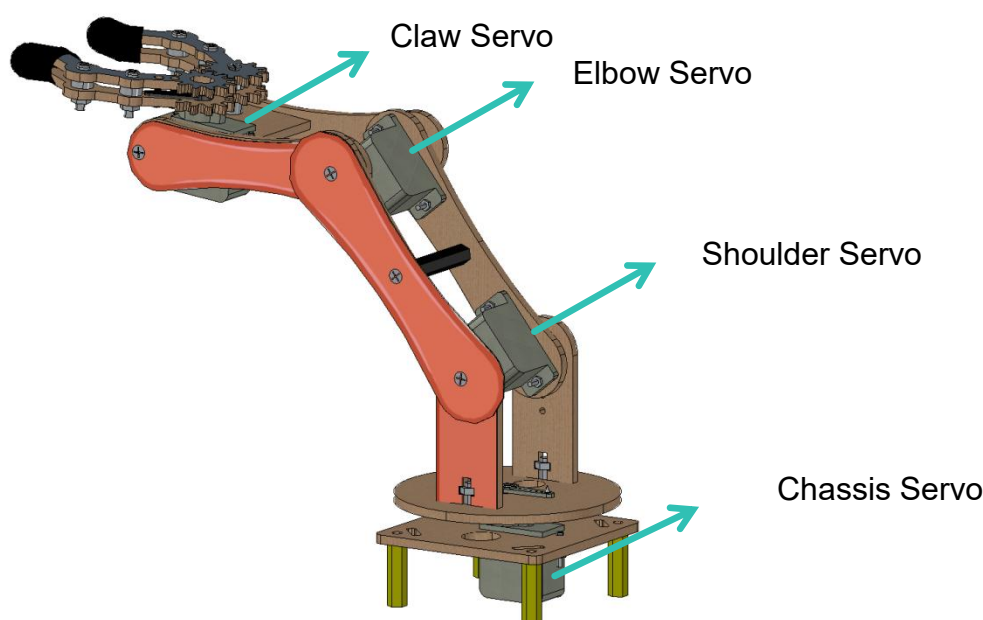


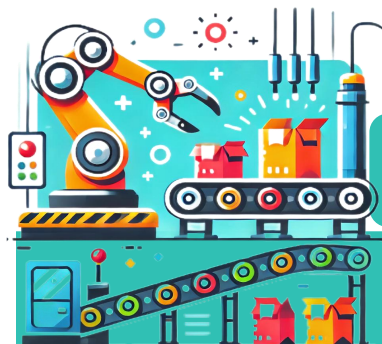
robotic arms can be used for precise planting and fertilizing, enhancing the level of automation in agricultural production.



RobotArm

The RobotArm used in this tutorial is a four-degree-of-freedom (4-DOF) robotic arm, with four servos driving the base, shoulder, elbow, and end-effector (hand gripper) joints. The servo that drives the base joint is called the base servo, the one that drives the shoulder joint is called the shoulder servo, the one that drives the elbow joint is called the elbow servo, and the one that drives the hand gripper is called the gripper servo.





Smart Factory Lab

Build Your Robotic Arm



Project Description

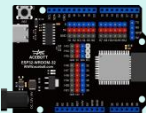
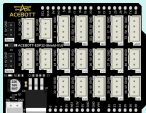


Follow the assembly steps in the building document to construct your own robotic arm.



Hardware Scheme

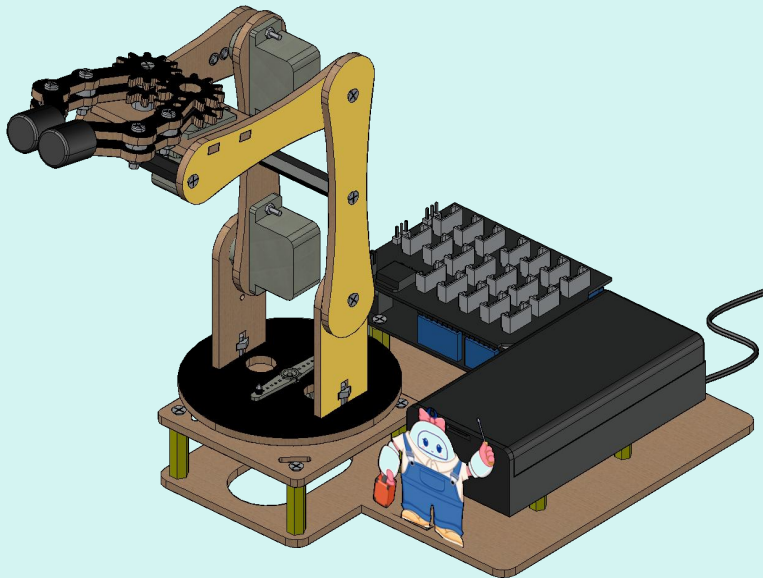
1. Hardware List

Picture	Name	Quantity
	ESP32 Board	1
	ESP32 Expansion Board	1
	MG90S Servo	4
	Type-C Data Cable	1



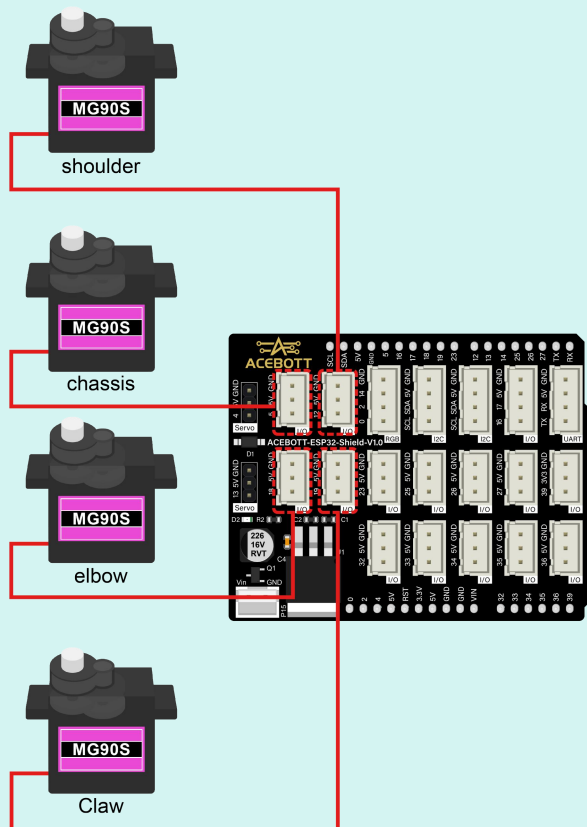


2. Hardware Structure



For detailed assembly steps, refer to Section 1 of the assembly document, titled "01 Build Your Robotic Arm."

3. Hardware Wiring



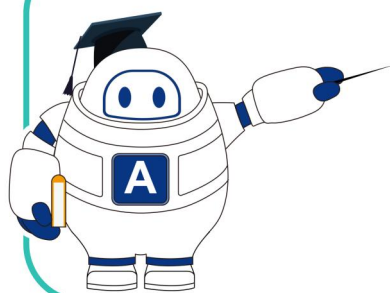


Section 2 Joint Motion of Robotic Arm



Course Objects

- Understand the working principle of a servo.
- Master the control methods of a servo.
- Understand the concept of variable.
- Learn how to use variable.
- Master the usage of the if statement.
- Learn how to control a robotic arm to grab an object.



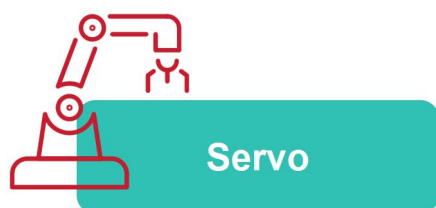
How to control the movements of a robotic arm?

The precise movements of a robotic arm rely on the coordinated work of its joints and links. Motors, acting as the power source, drive the joints to rotate flexibly, which in turn moves the links accordingly. This chain reaction ensures the accurate positioning of the robotic arm's end effector, allowing it to reach the designated target location precisely. Therefore, controlling the movement of





a robotic arm means controlling the movement of the motors on the arm. Depending on different functional requirements, the types of motors on a robotic arm may vary. In this tutorial, the motors driving the robotic arm are servos.



A servo is a device that can precisely control position, speed, and acceleration, widely used in robotics, models, and automation control systems. A servo typically consists of a motor, a gear set, a control circuit, and a position sensor (usually a potentiometer).

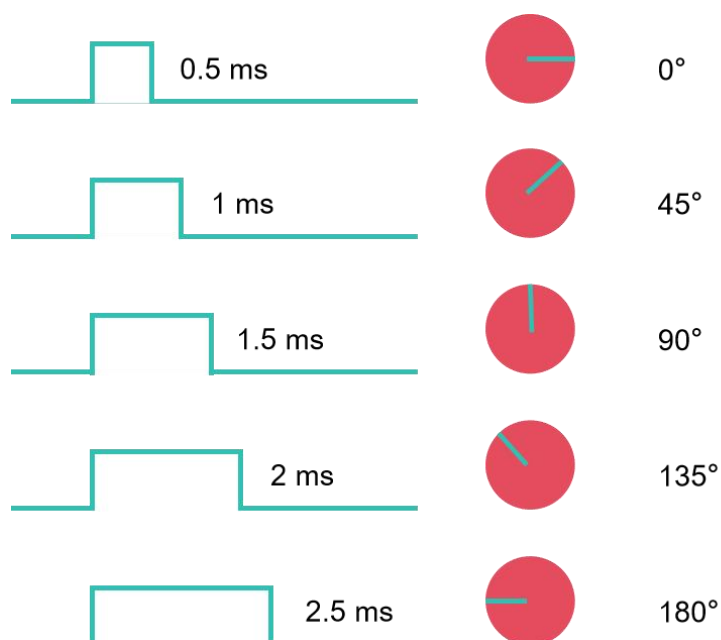
1. **Motor:** The motor inside the servo is its power source, typically using a DC motor to provide rotational power.
2. **Gear Set:** The gear set converts the motor's high-speed, low-torque output into low-speed, high-torque output, allowing the servo to achieve precise angle control.
3. **Position Sensor:** The position sensor is used to detect the current angle of the servo. The potentiometer provides a voltage signal corresponding to the rotational angle through changes in resistance.





4. Control Circuit: The control circuit is the core part of the servo, responsible for receiving control signals and driving the motor. The control circuit typically includes a pulse-width modulation (PWM) receiver and a feedback control system. By comparing the signal from the position sensor with the target position signal, the control circuit generates an error signal and adjusts the motor's rotation direction and speed based on this error.

Servos are typically controlled using a PWM signal. The PWM signal is a pulse signal, and its duty cycle (the ratio of the high-level time to the total period time) determines the target position of the servo. The common PWM signal frequency is 50Hz (20ms period), with a pulse width varying between 0.5ms to 2ms, corresponding to a rotation range of 0 degrees to 180 degrees.

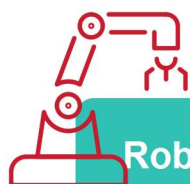


The servo model used in this tutorial is the MG90S, with a motion range of 0°-180° for the servo shaft. It has three colored connection wires: orange, red,





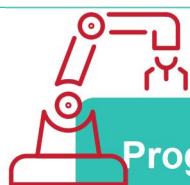
and brown. The orange wire connects to the I/O pin of the ESP32, the red wire connects to 5V, and the brown wire connects to GND.



RobotArm Joint Control

There is a precise correlation between the joint movements of the robotic arm and the control of the servos, ensuring that each joint's movement is closely linked to its corresponding driving servo. Here is a detailed explanation of this relationship:

Joint	Movement Pattern
Chassis	When the servo angle increases, the base moves to the left.
Shoulder	When the servo angle increases, the shoulder link moves downward.
Elbow	When the servo angle increases, the elbow link moves upward.
Claw	When the servo angle increases, the gripper opens.



Programming Preview

In the programming practice of this lesson, we will use variable and if statements.

1. Variable



In ACECode programming, the usage of variables is as follows:

New Variable

New variable name:

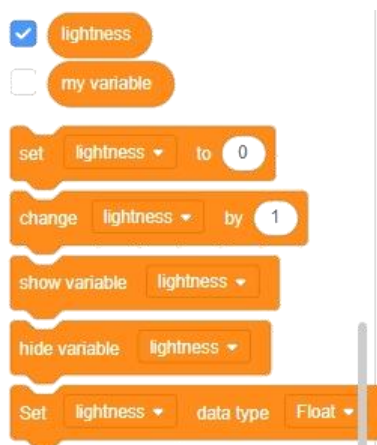
☒ For all sprites

☐ For this sprite only

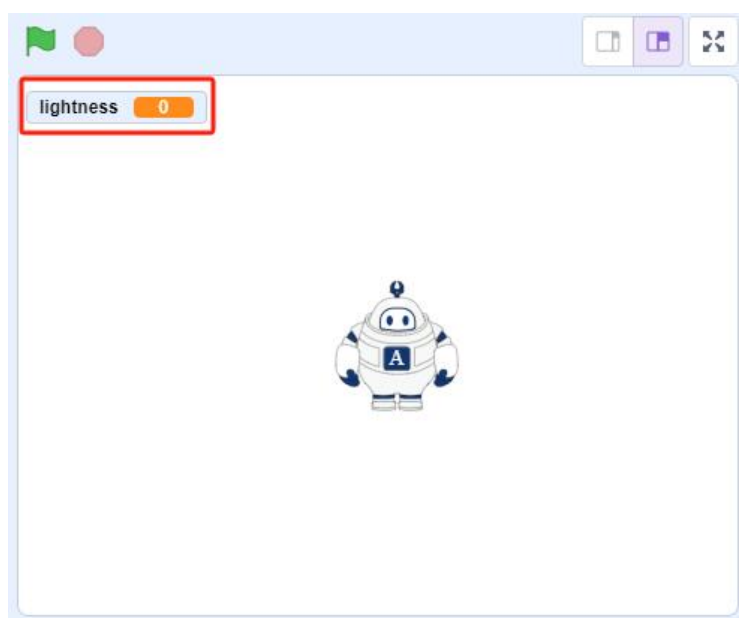
Cancel

OK

42



Additionally, in "online mode," you can display the current value of variables to facilitate debugging.



Data Type: When creating variables, it is necessary to declare the data type that the variable will store. In the ACECode programming environment, basic data types include integer (integer), float(float), character (char), and string (string), etc. If the variable data type is not set in the program, it defaults to integer.





Variable Name: The variable name represents the meaning of the data stored in the variable and its location in memory. The variable name allows you to access and use these data in subsequent parts of the program.

Variable Value: This refers to the data stored in a variable. The value of a variable can be updated and modified at any time to meet the needs of program execution.

2. if

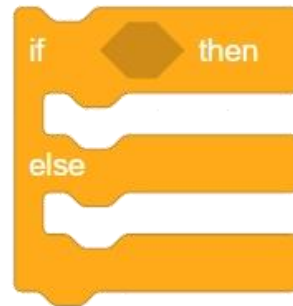
The 'if' statement is a fundamental branching structure in programming. It executes different blocks of code based on the truth value of a condition, allowing the program to make decisions based on specific criteria, and thus flexibly implement the logic of conditional branching. In ACECode, depending on the number of conditions, the 'if' statement is divided into two structures: 'if' and 'if...else....' The usage of the 'if' statement is as follows:

(1) 'if' Statement Instruction: When the condition in the hexagonal box is true, execute the code block contained within the 'if' statement; otherwise, skip the code block.

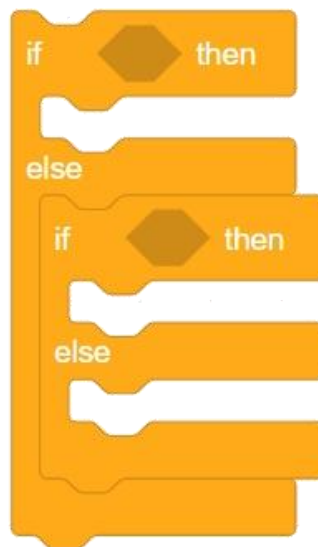


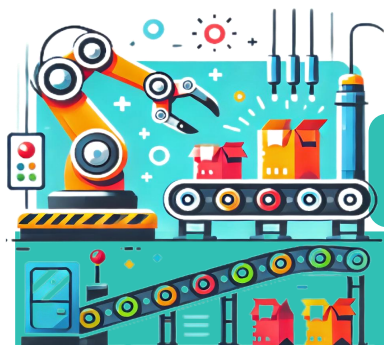


(2) The 'if...else...' statement instruction: The 'if...else...' statement is used for two opposing conditional scenarios. When the condition is true, execute the code block contained under 'then'; otherwise, execute the code block contained under 'else'.



(3) If there are multiple conditions to be evaluated, multiple 'if...else...' statements can be combined and used together.





Smart Factory Lab

Joint Motion of Robotic Arm



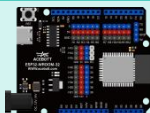
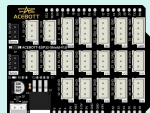
Project Description

Input a command from the keyboard to control the movement of the robotic arm's joints to the corresponding position.



Hardware Scheme

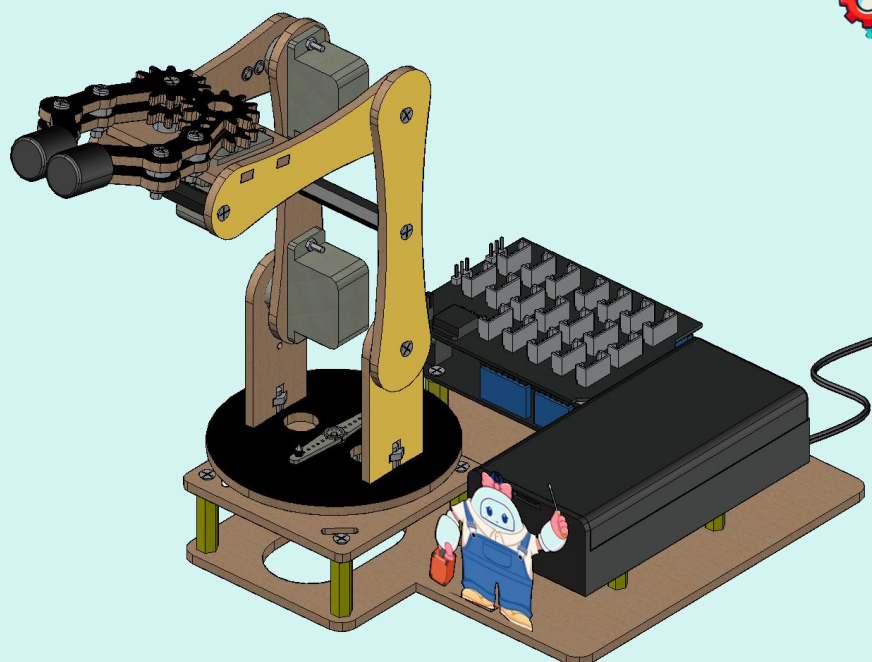
1. Hardware List

Picture	Name	Quantity
	ESP32 Board	1
	ESP32 Expansion Board	1
	MG90S Servo	4
	Type-C Data Cable	1



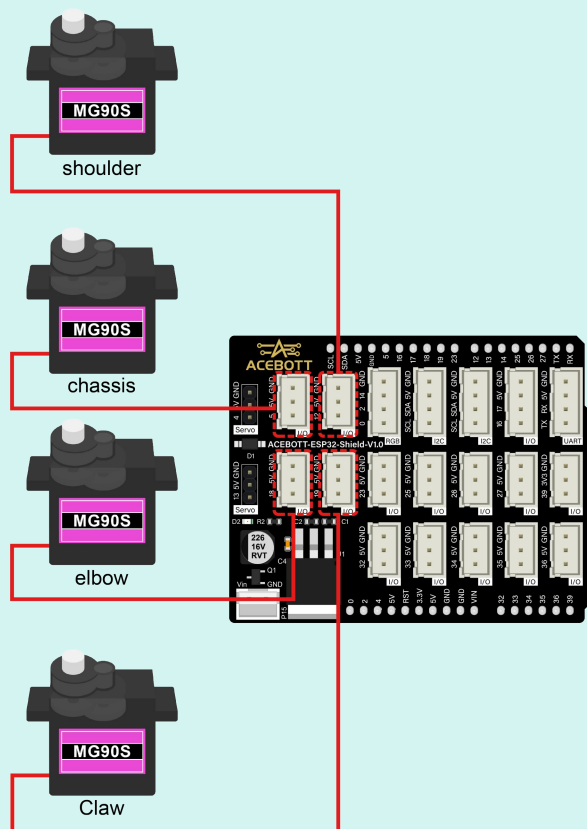


2. Hardware Structure



For detailed assembly steps, refer to Section 2 of the assembly document, titled "02 Joint Motion of Robotic Arm."

3. Hardware Wiring

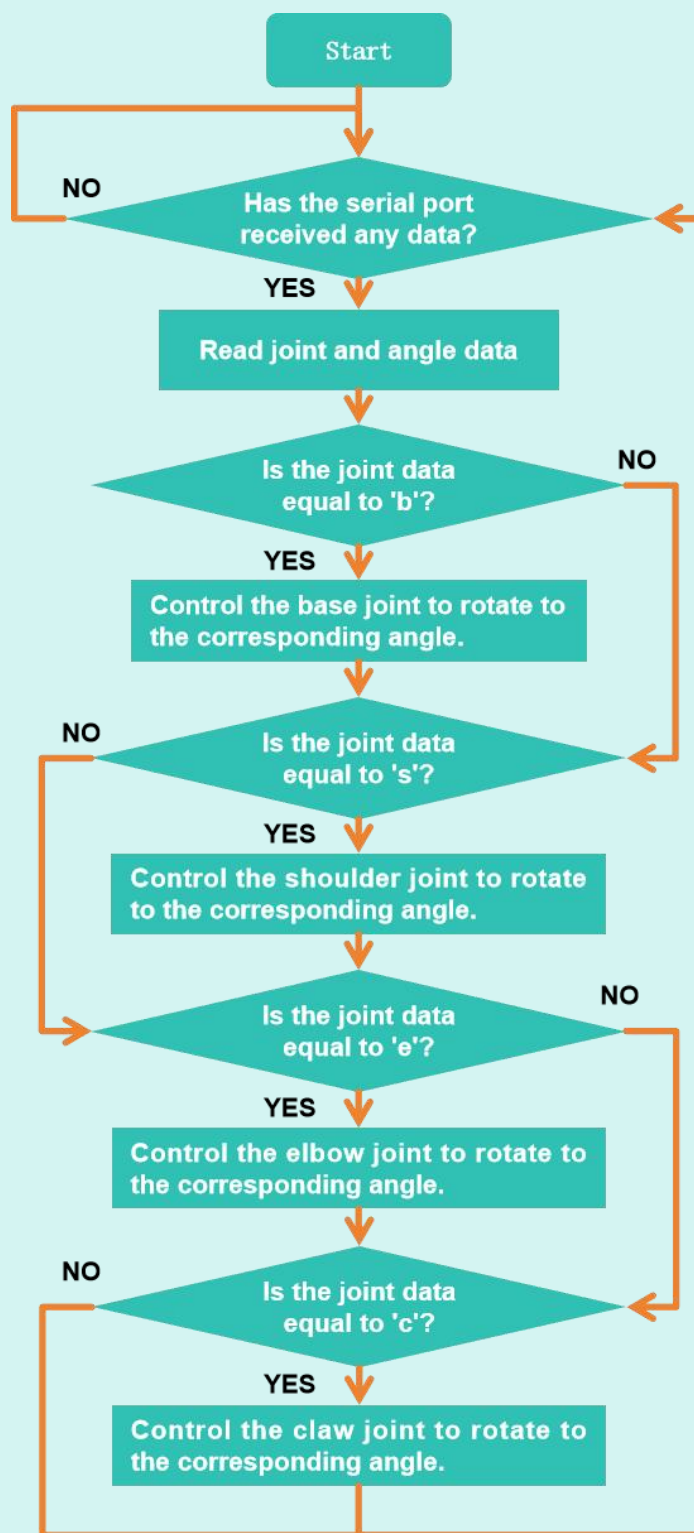




Programming

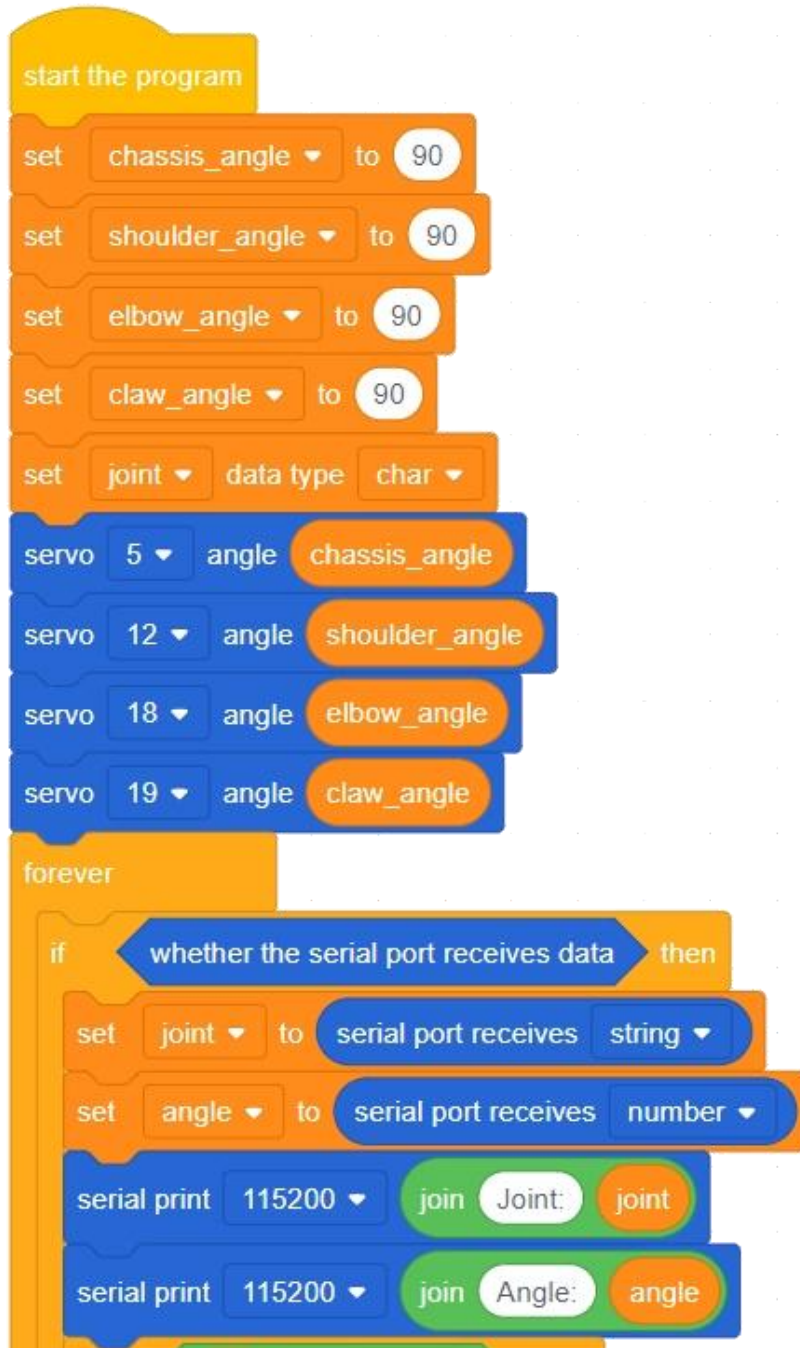


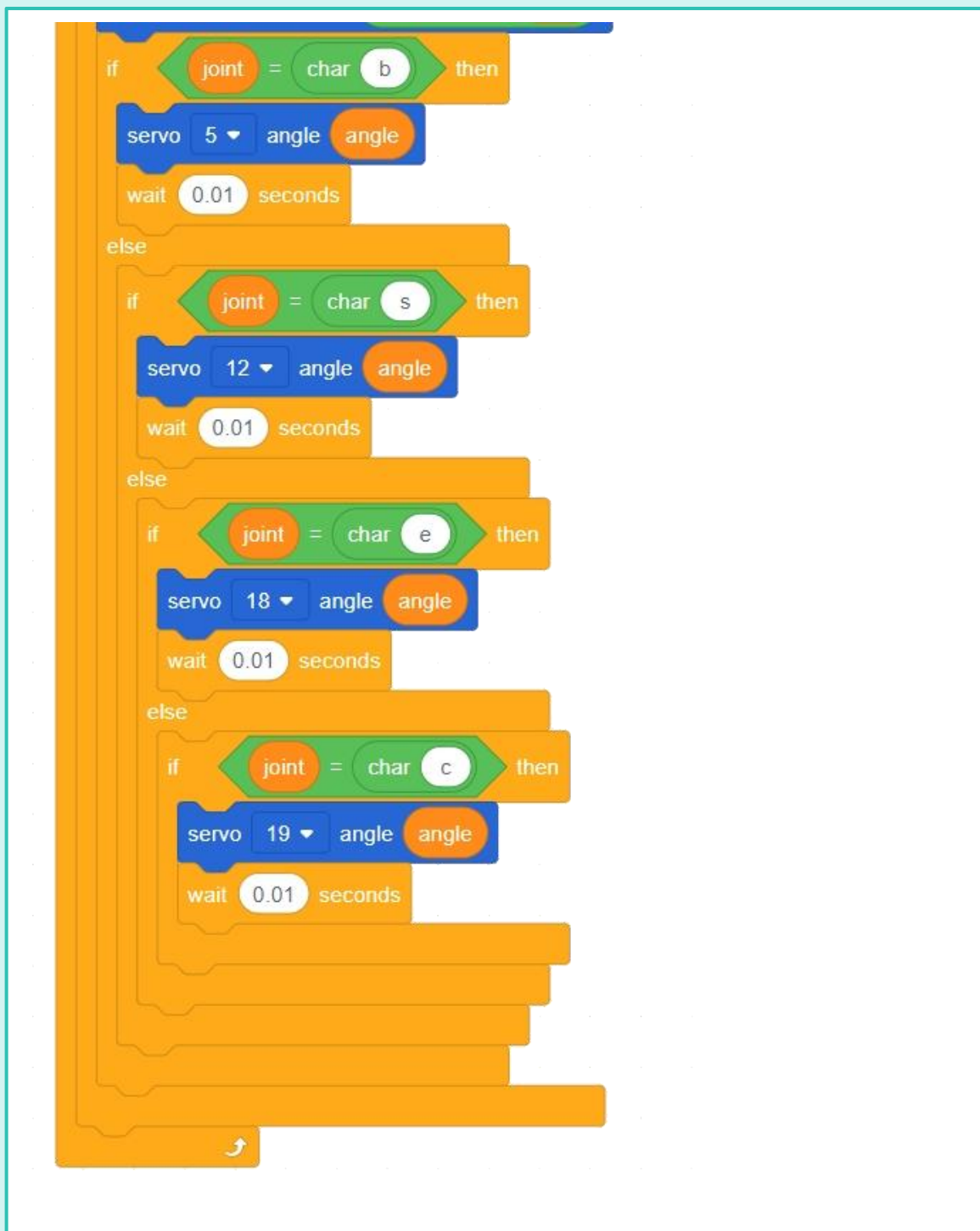
1. Program Flowchart





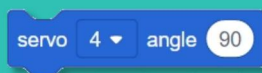
2. Coding







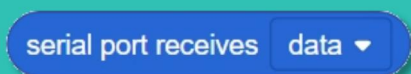
3. Instruction Parsing



: This instruction is in the "Robots" command module. It controls the servo motor's shaft to rotate to a specified angle position. The drop-down box allows you to select the corresponding pin, and the input box is for the target value to which the servo motor should rotate.

whether the serial port receives data

: This instruction is within the "Robots" command module. It can determine whether data is received through the serial port.

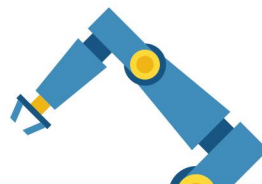


: This instruction is within the "Robots" command module. It allows you to retrieve data received through the serial port. By clicking on the drop-down box, you can obtain different types of data input from the serial port.



: This instruction is in the "Operators" command module. It is one of the relational operators, which returns true when the values on its left and right satisfy this relationship, and false otherwise.



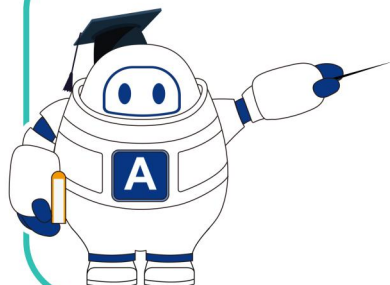


Section 3 Speed Control of Robotic Arm



Course Objects

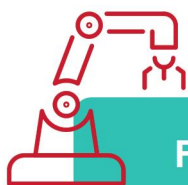
- Understand the factors affecting servo motor speed.
- Understand the concept of iteration.
- Master the use of “repeat until...”.
- Master the methods for controlling the speed of the robotic arm.



How to control the speed of robotic arm?

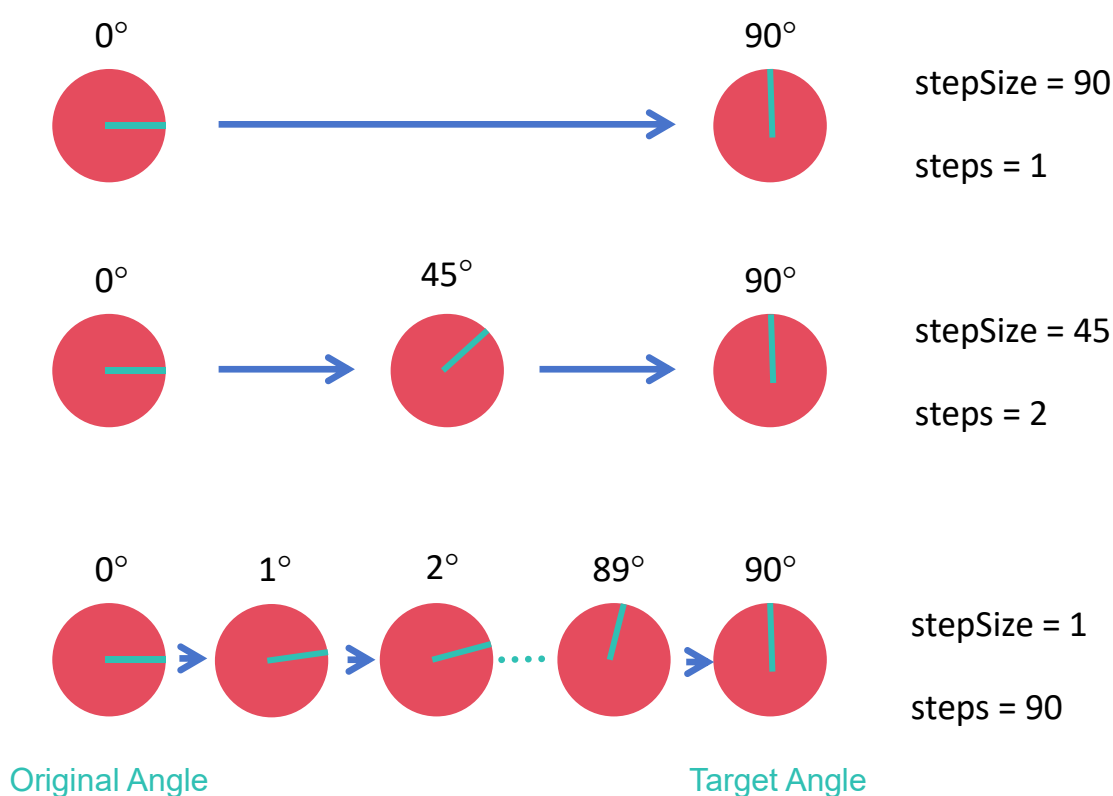
In the previous lesson, we successfully got the robotic arm to start operating. However, during its motion, it exhibited high speed and significant inertia, which made the arm's operation unstable. To achieve more precise control, we need to finely adjust the movement speed of the robotic arm. Since the arm's movement is driven by servos on the joints, controlling the arm's speed essentially means adjusting the rotation speed of the servos.





Factors Affecting Servo Speed

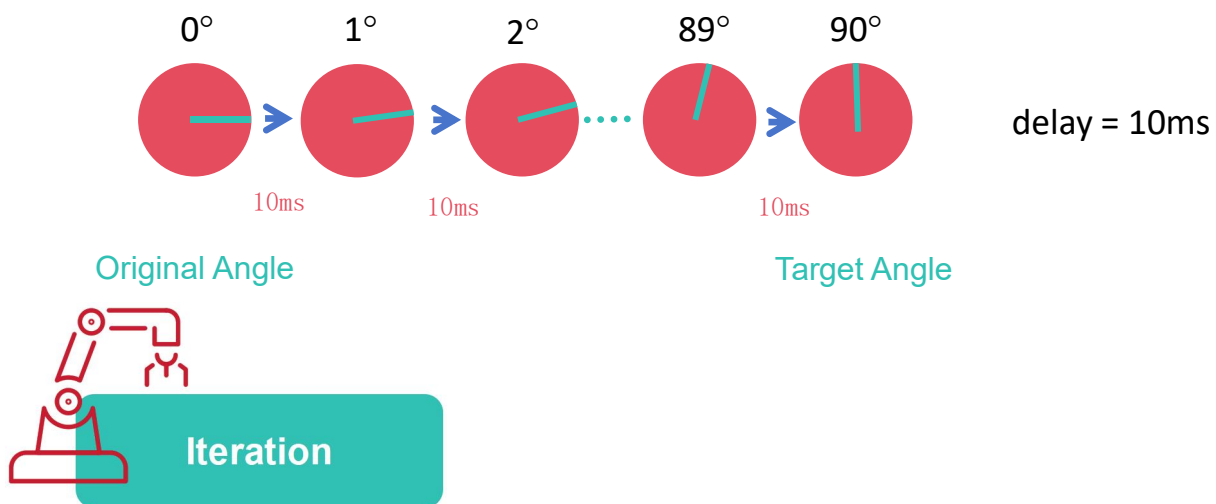
The motion process of a servo involves rotating the servo shaft from an initial angle to a target angle. To slow down this motion, we can divide it into multiple steps and execute them progressively. The number of steps depends on the step size setting; specifically, the smaller the step size, the more steps are required, resulting in a smoother and slower movement of the servo to reach the target angle.



The rotation speed of a servo is not only determined by the step size but also significantly influenced by the time interval between each step. As the time

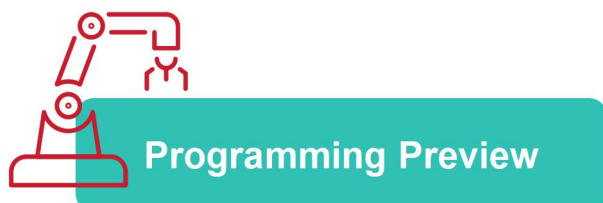


interval increases, the rotation speed of the servo decreases correspondingly, while a shorter time interval results in a faster rotation speed.



The process of servo stepwise movement is essentially an iterative process. Each step involves rotating by a fixed step size from the previous step, and this process is repeated until the servo reaches the target angle.

Iteration is the activity of repeating a feedback process with the aim of approaching a desired target or result. Each repetition of the process is referred to as an "iteration," and the result obtained from each iteration serves as the initial value for the next iteration.



In ACECode programming, we can use the "repeat...until" command to implement iterative algorithms. "Repeat...until" is a loop structure

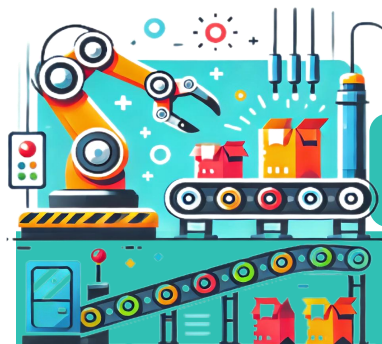


command used to repeatedly execute a block of code. Its basic usage is as follows:



Hexagonal input box: The condition expression input box checks the value of the expression in the box before each loop iteration. If the value is false, the loop continues to execute the code contained within the command. If the value is true, the loop terminates.





Smart Factory Lab

Control Speed of Robotic Arm



Project Description

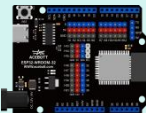
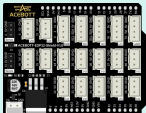


Control the rotation speed of the robotic arm to make its movement smoother and more fluid.



Hardware Scheme

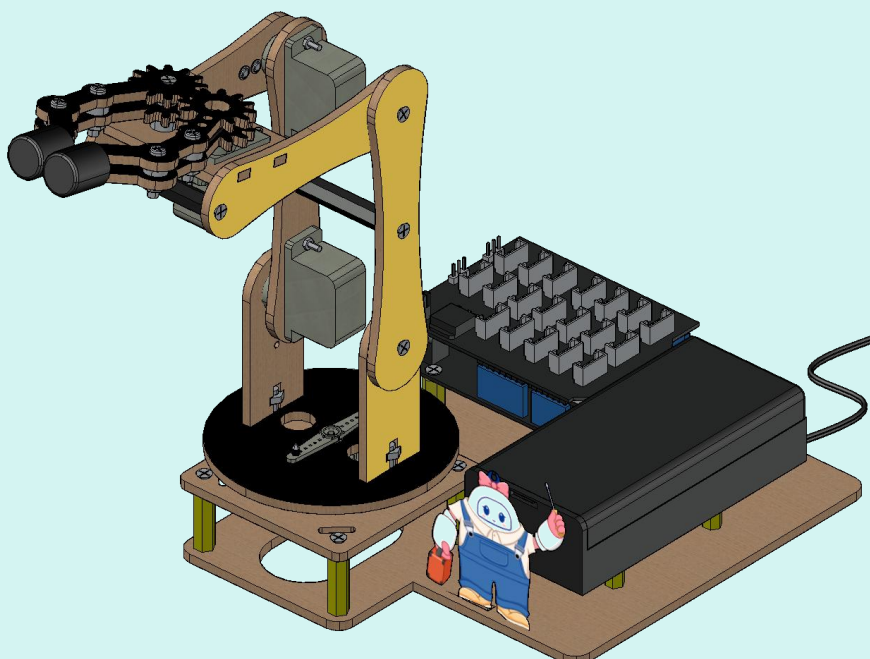
1. Hardware List

Picture	Name	Quantity
	ESP32 Board	1
	ESP32 Expansion Board	1
	MG90S Servo	4
	Type-C Data Cable	1



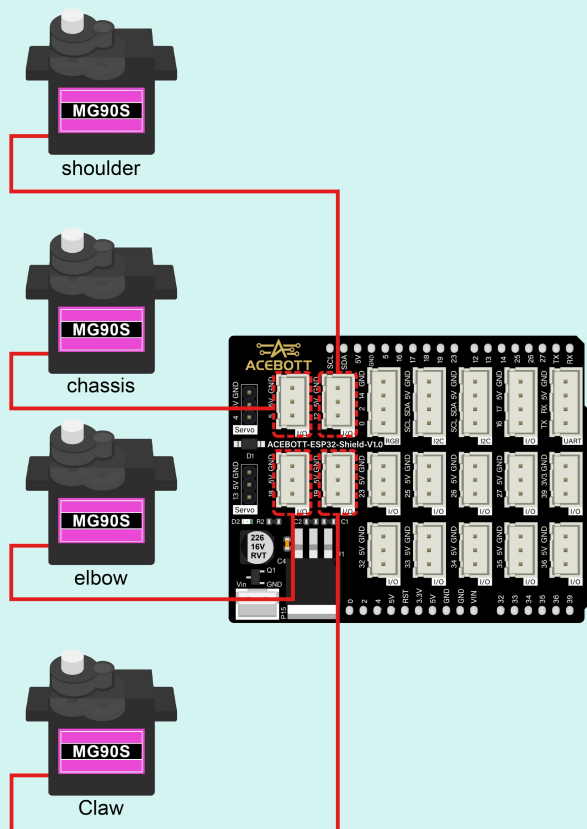


2. Hardware Structure



For detailed assembly steps, refer to Section 3 of the assembly document, titled "03 Control Speed of Robotic Arm."

3. Hardware Wiring

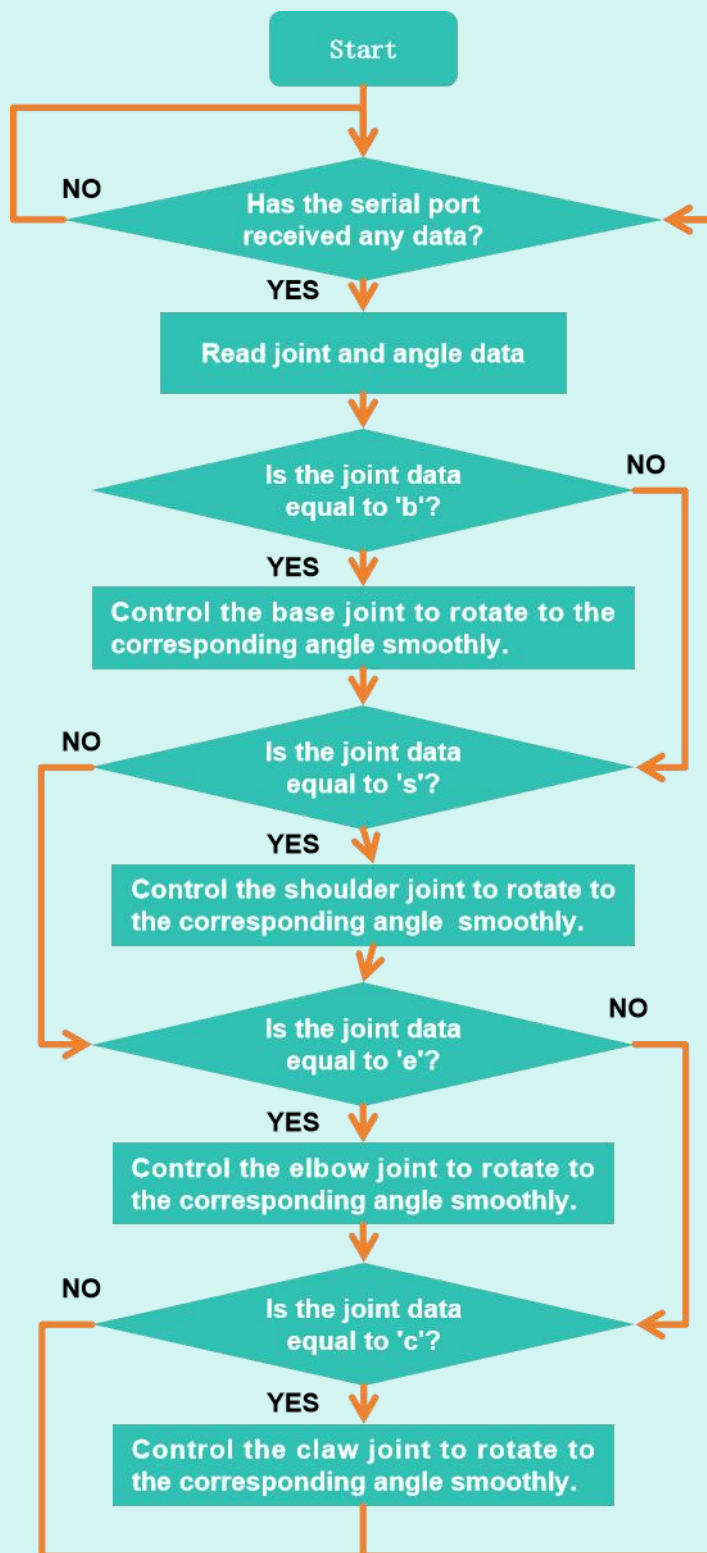




Programming

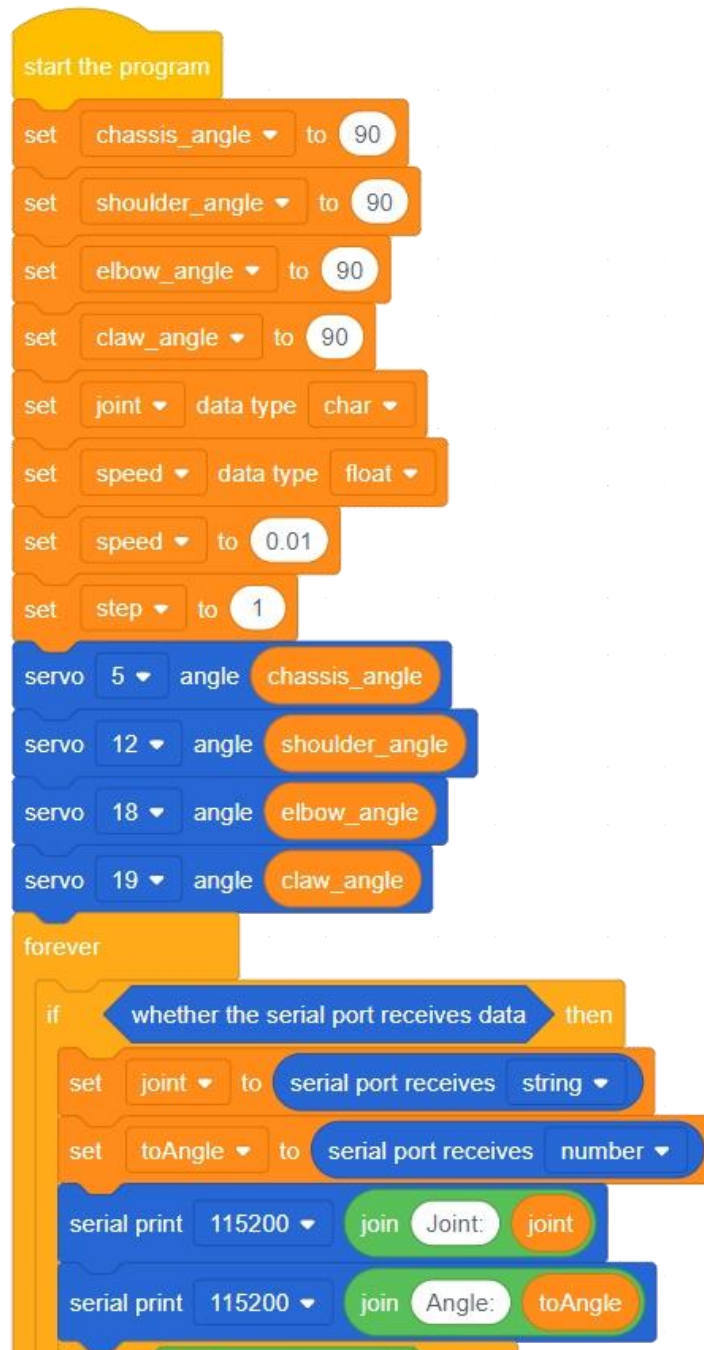


1. Program Flowchart





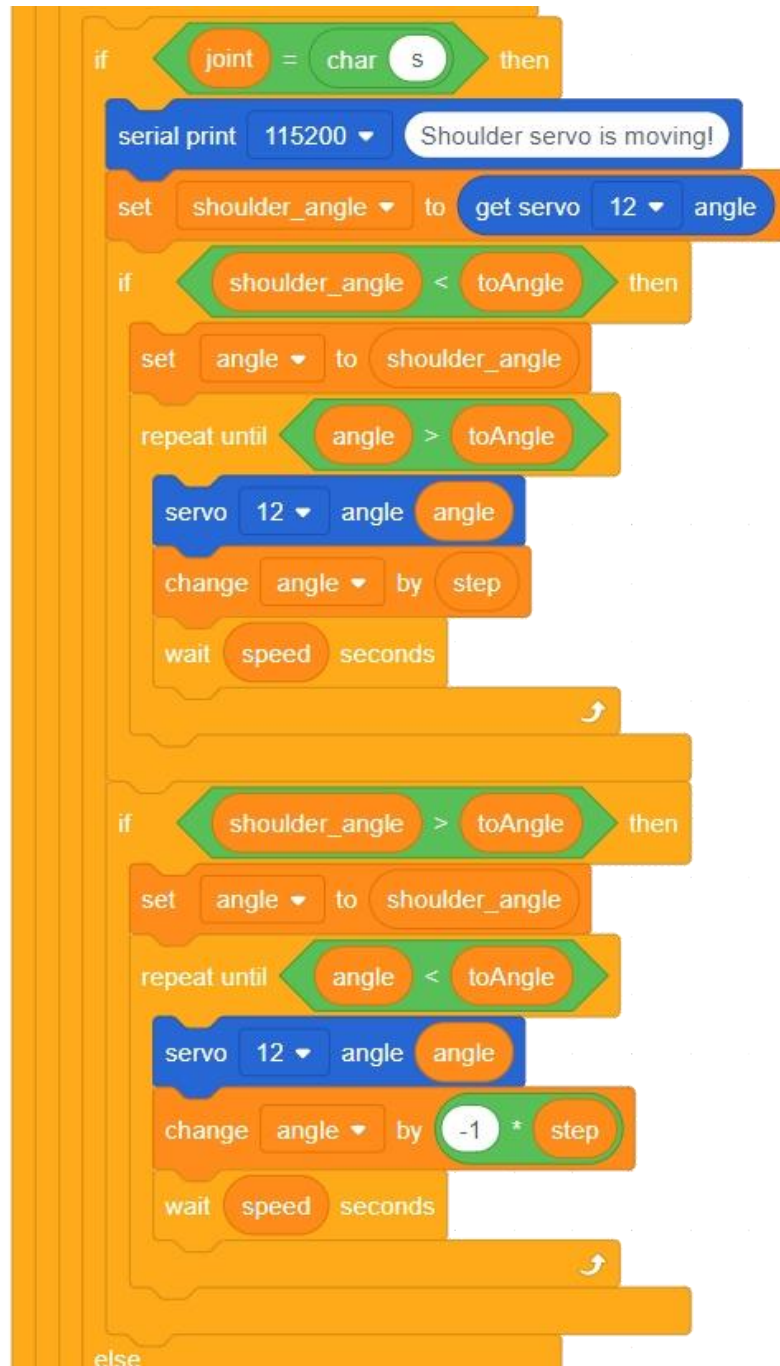
2. Coding

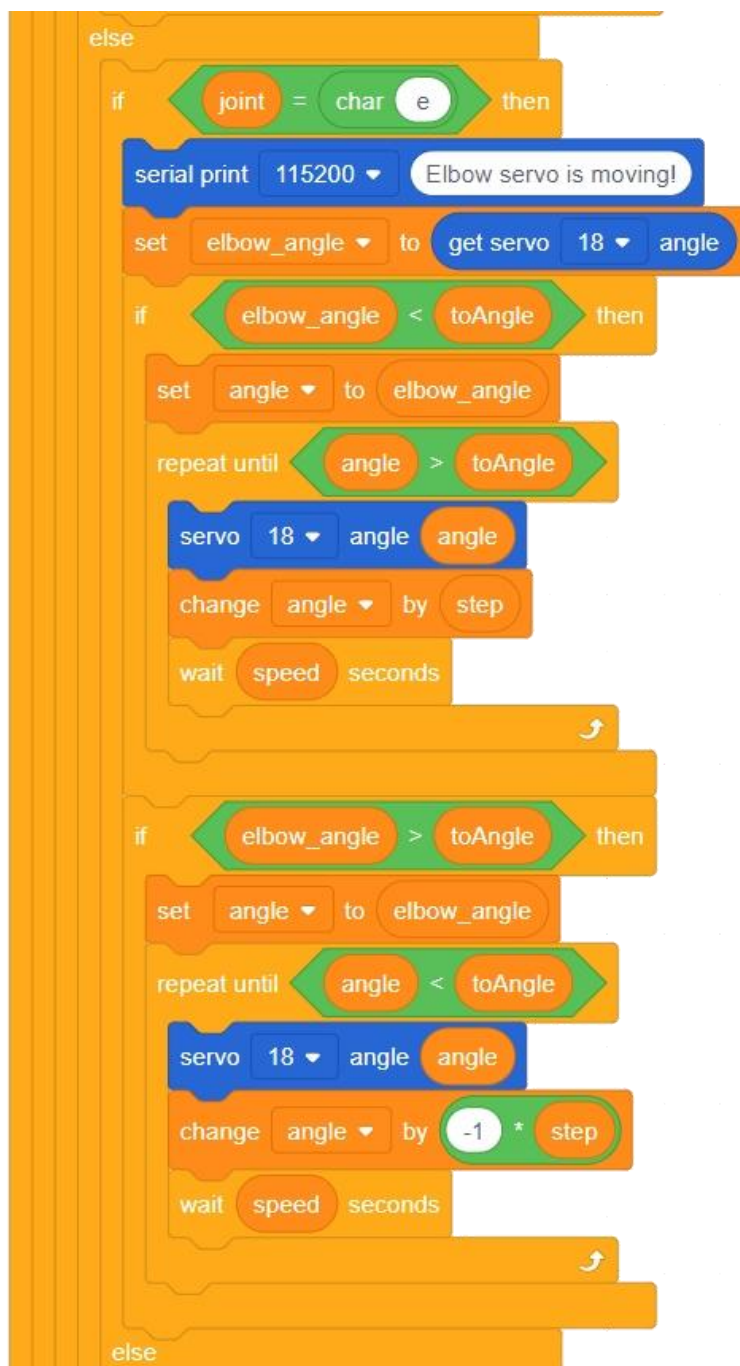


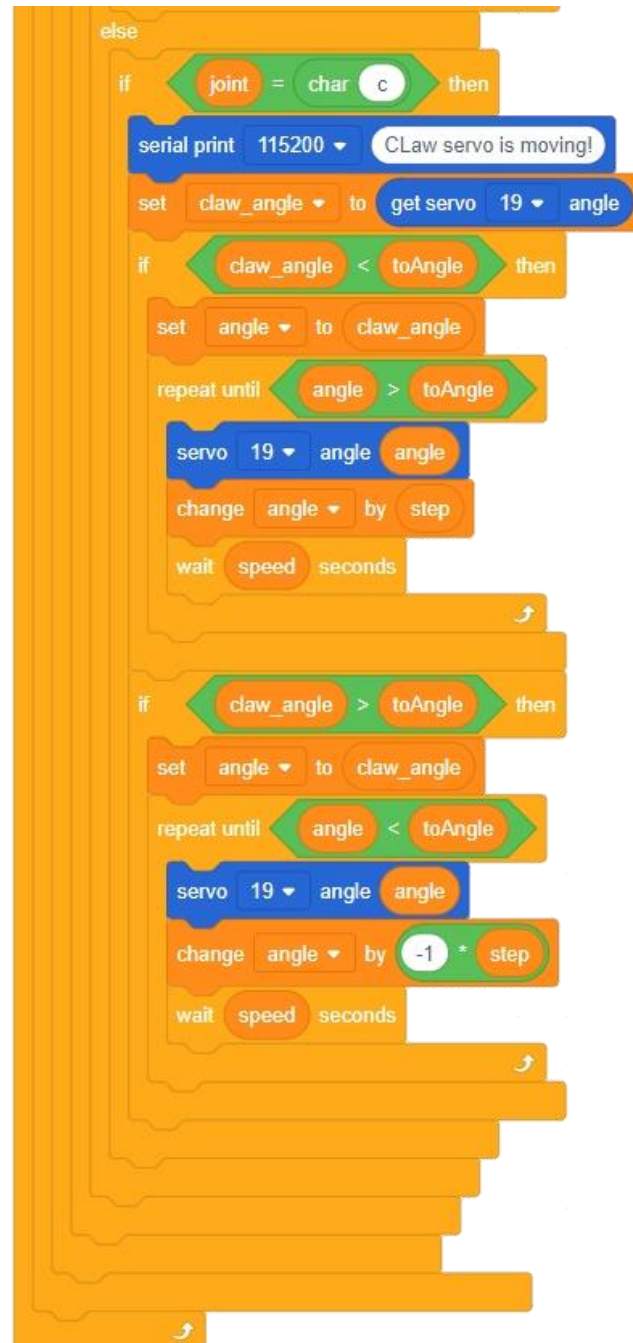


```
if joint = char b then
  serial print 115200 Chassis servo is moving!
  set chassis_angle to get servo 5 angle
  if chassis_angle < toAngle then
    set angle to chassis_angle
    repeat until angle > toAngle
      servo 5 angle angle
      change angle by step
      wait speed seconds
  if chassis_angle > toAngle then
    set angle to chassis_angle
    repeat until angle < toAngle
      servo 5 angle angle
      change angle by -1 * step
      wait speed seconds
  else
```









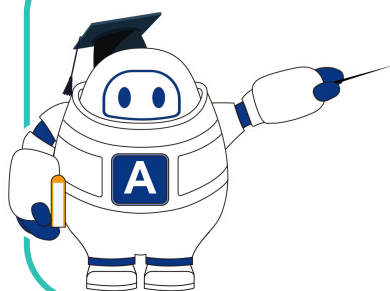


Section 4 Smart Handling



Course Objects

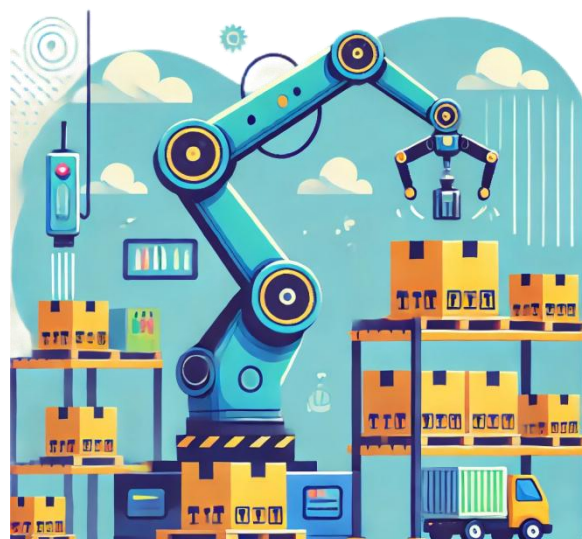
- Master the method of using a robotic arm to handle objects.
- Understand the concept of function.
- Master the usage of functions.



What can a robotic arm do in smart factory?

In a factory, robotic arms can perform various tasks to enhance production efficiency, precision, and safety. Here are some common applications:

Assembly: Robotic arms can assemble various components, placing them accurately in the correct positions. This is especially common in electronics and automotive manufacturing.





Welding: In automotive and metal processing industries, robotic arms can perform welding tasks, ensuring the quality and consistency of weld points.

Handling: Robotic arms can move heavy or hazardous items, reducing the labor intensity and risk for workers. This is widely used in warehousing and logistics.

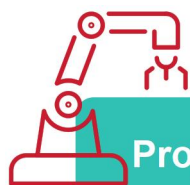
Packaging: In the food, beverage, and consumer goods industries, robotic arms can efficiently package products, enhancing packaging speed and accuracy.

Quality Inspection: Robotic arms can be equipped with sensors and cameras to inspect product quality, detecting defects and flaws.

Cutting and Processing: Robotic arms can carry out laser cutting, milling, and drilling operations, suitable for processing metals, plastics, and other materials.

In this lesson, we will focus on enabling the robotic arm to perform automatic handling tasks. During the handling process, the end of the robotic arm needs to reach multiple points. To simplify the programming of the handling task, we will encapsulate the code for controlling the robotic arm's joints into functions.

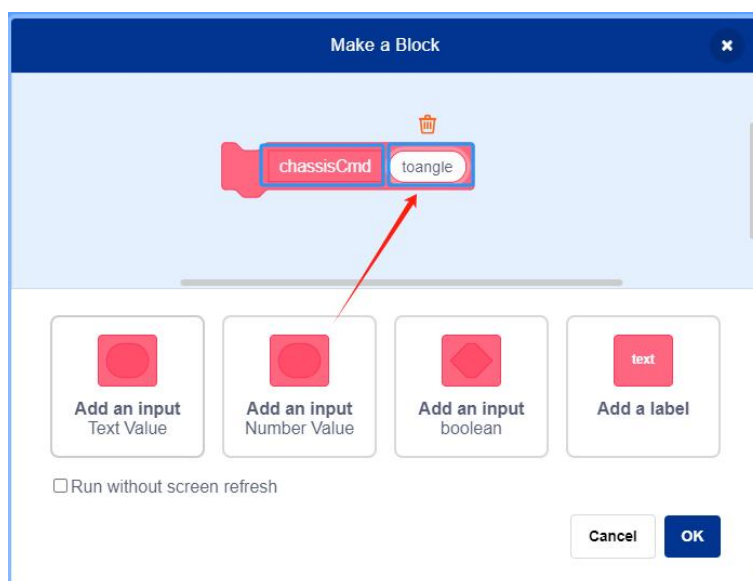




Programming Preview

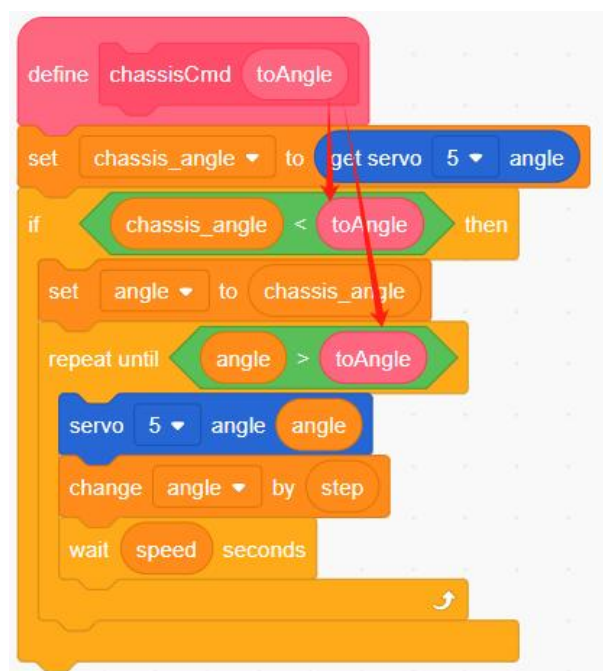
In ACECode, you can encapsulate a block of code that can perform a specific task and can be reused into a block, and then call the code encapsulated within it by using a My Block in the main program. This can improve the readability of the program and the efficiency of programming. In programming, it is also referred to as a function.

You can create a My Block from the "My Block" instruction category by clicking "Make a Block." First, define a clear and meaningful name for it; then add parameters based on the actual function of the my block. If no input parameters are needed, you can omit them. For example, here we define a block to control the chassis servo to rotate to a specified angle, named "chassisCmd," and add a "number value" parameter to it.

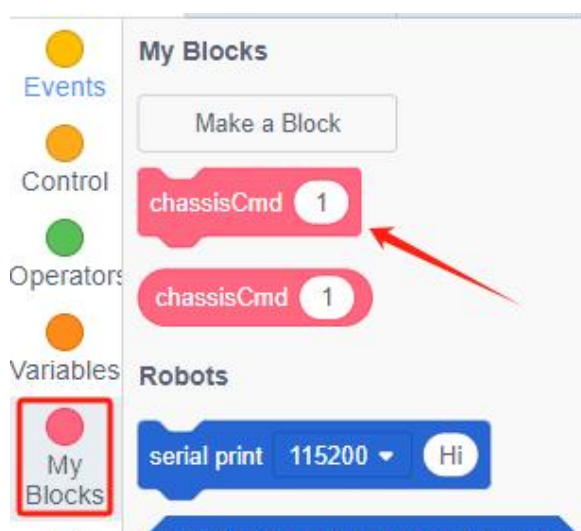


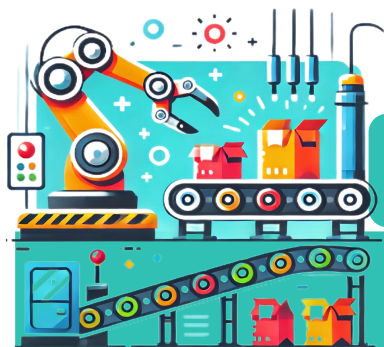


After creating a block, a "Define chassisCmd" block will appear in the programming area, where you can write the code contained in the my block. If you need to use its parameters, you can drag the parameters down from the block definition to use them.



When you want to use a my block that you have already written, you can drag it out from the "My Blocks" instruction module for use.





Smart Factory Lab

Smart Handling



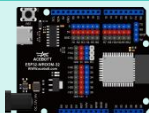
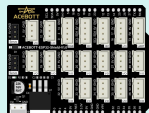
Project Description

Control the robotic arm to automatically transport an object from point A to point B.



Hardware Scheme

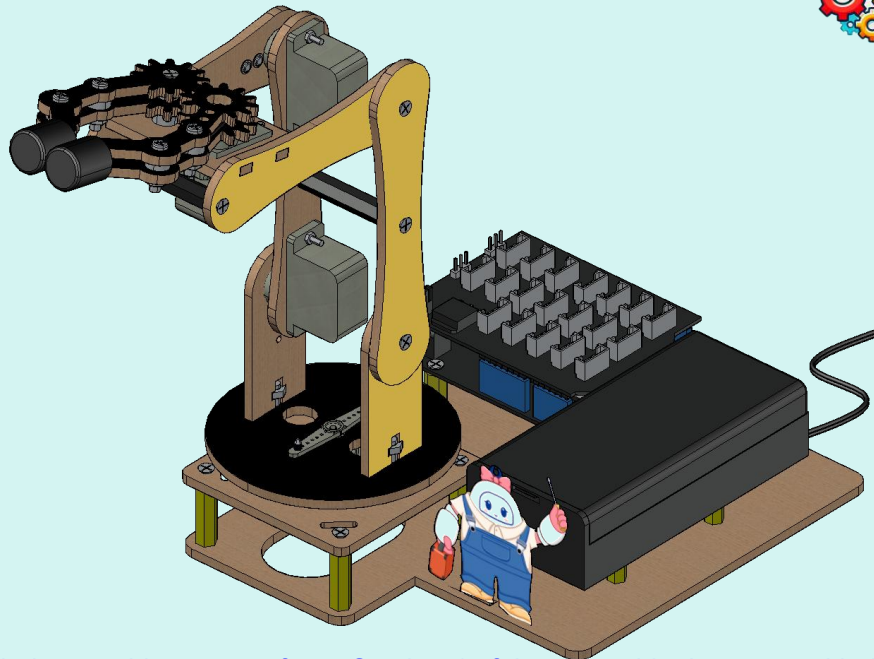
1. Hardware List

Picture	Name	Quantity
	ESP32 Board	1
	ESP32 Expansion Board	1
	MG90S Servo	4
	Type-C Data Cable	1



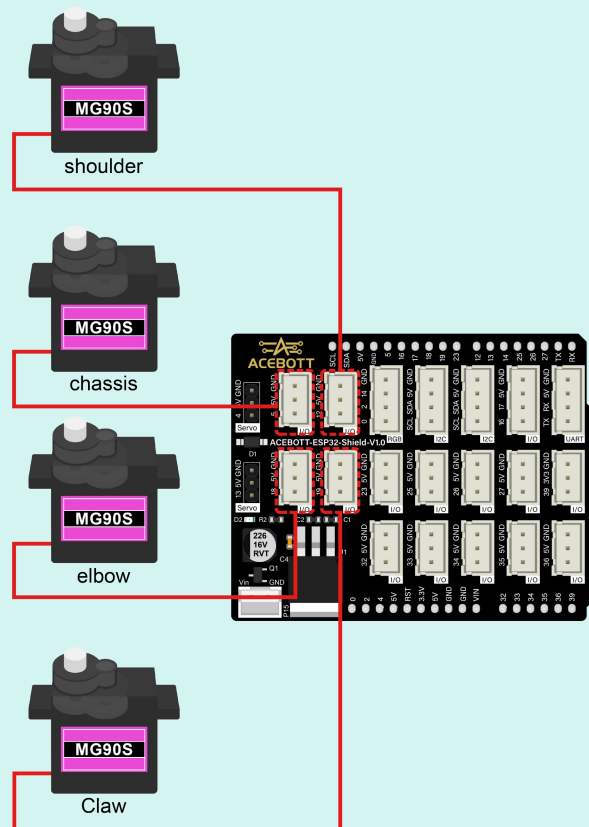


2. Hardware Structure



For detailed assembly steps, refer to Section 4 of the assembly document, titled "04 Smart Handling."

3. Hardware Wiring

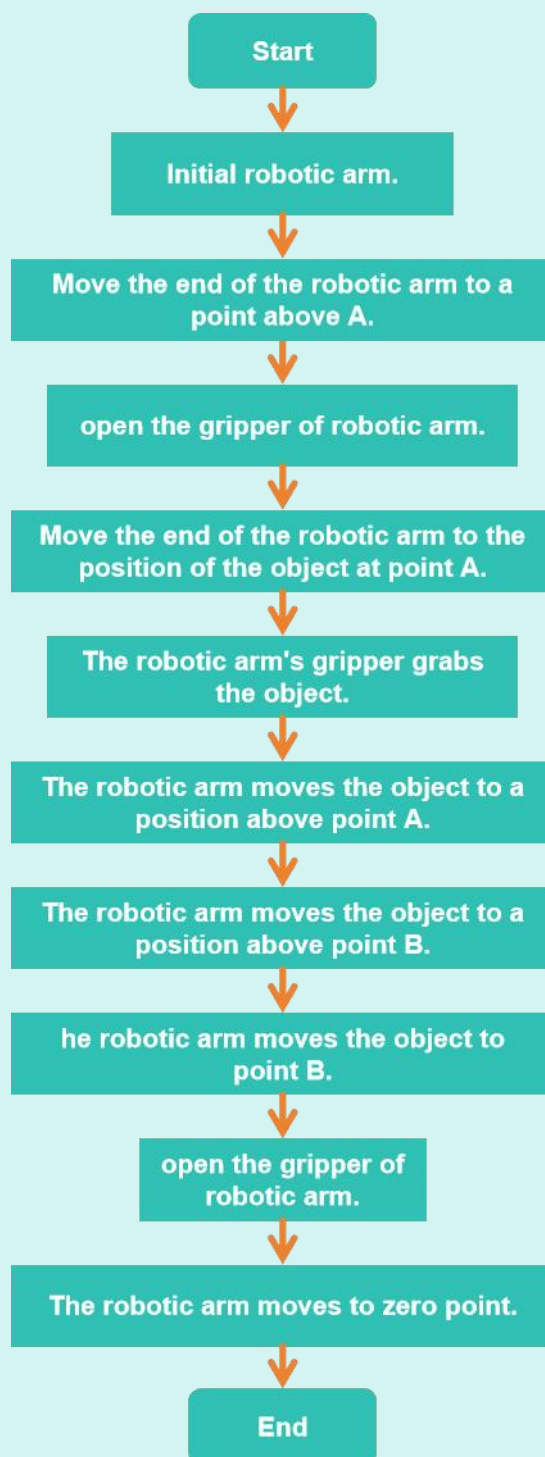




Programming



1. Program Flowchart

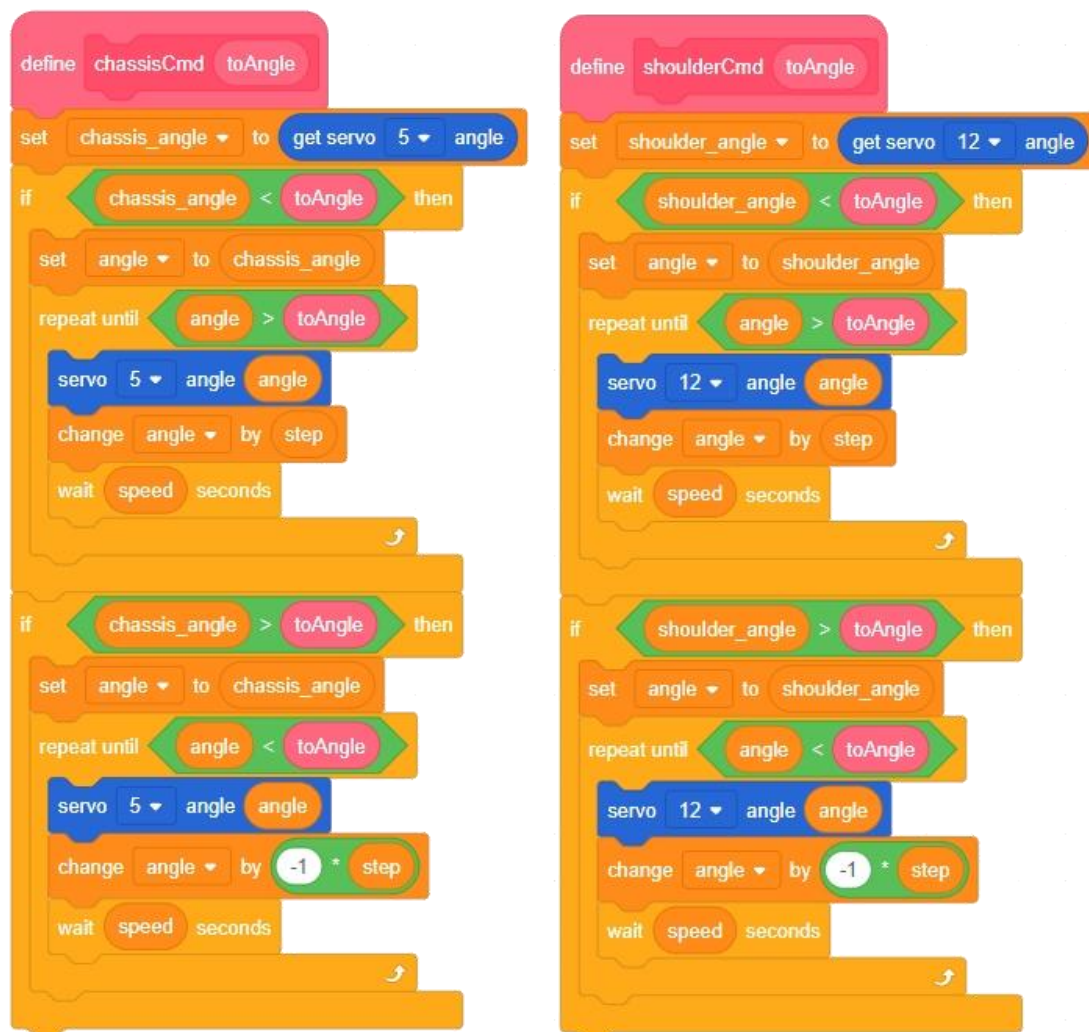




2. Coding

```
start the program
set chassis_angle to 90
set shoulder_angle to 90
set elbow_angle to 90
set claw_angle to 90
set joint data type char
set speed data type float
set speed to 0.01
set step to 1
servo 5 angle chassis_angle
servo 12 angle shoulder_angle
servo 18 angle elbow_angle
servo 19 angle claw_angle
clawCmd 120
chassisCmd 135
shoulderCmd 135
elbowCmd 57
clawCmd 90
elbowCmd 90
chassisCmd 50
elbowCmd 57
clawCmd 120
chassisCmd 90
shoulderCmd 90
elbowCmd 90
clawCmd 90
```







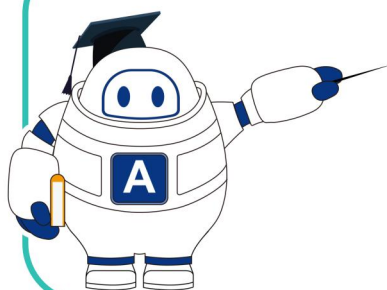


Section 5 Robotic Arm Coordinate Control



Course Objects

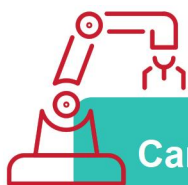
- Understand the concept of the Cartesian coordinate system.
- Familiarize the methods for representing the end-effector's spatial coordinates of a robotic arm.
- Understand the concepts of forward and inverse kinematics.
- Master the use of spatial coordinates to control the motion of the robotic arm.



How to determine the position of the robotic arm's end effector in space?

To determine the position of a point in space, a reference coordinate system is typically used. Common coordinate systems include Cartesian coordinates, polar coordinates, cylindrical coordinates, and spherical coordinates. In practice, the Cartesian coordinate system is most commonly used. In this lesson, we will focus on the motion control of robotic arms within the Cartesian coordinate system.



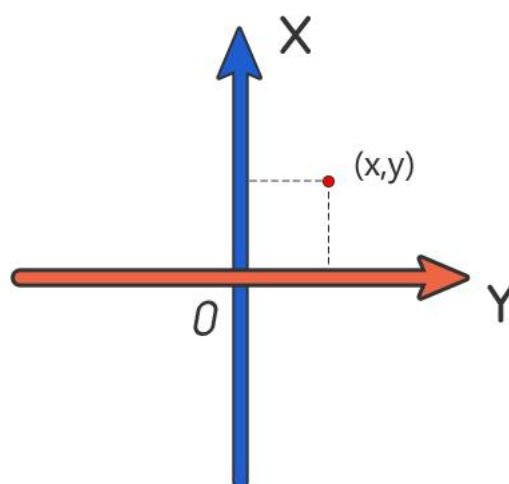


Cartesian Coordinate System

The Cartesian coordinate system is a mathematical system used to describe the position of points in two-dimensional or three-dimensional space. It consists of perpendicular coordinate axes through which the position of each point can be defined.

1. Two-Dimensional Cartesian Coordinate System

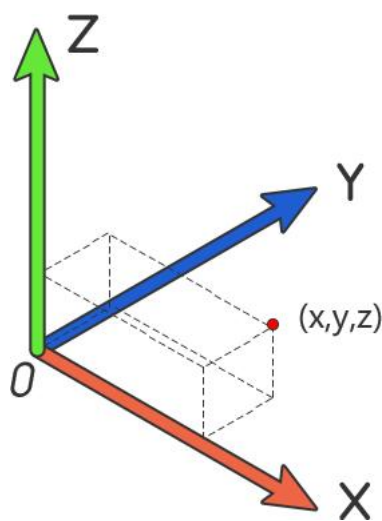
The two-dimensional Cartesian coordinate system is composed of two perpendicular axes: the X-axis (horizontal axis) and the Y-axis (vertical axis). The point where these axes intersect is called the origin, denoted as O . In the two-dimensional space of this coordinate system, any point can be represented by (x, y) , where x is the position of the point's projection on the X-axis, and y is the position of the point's projection on the Y-axis.





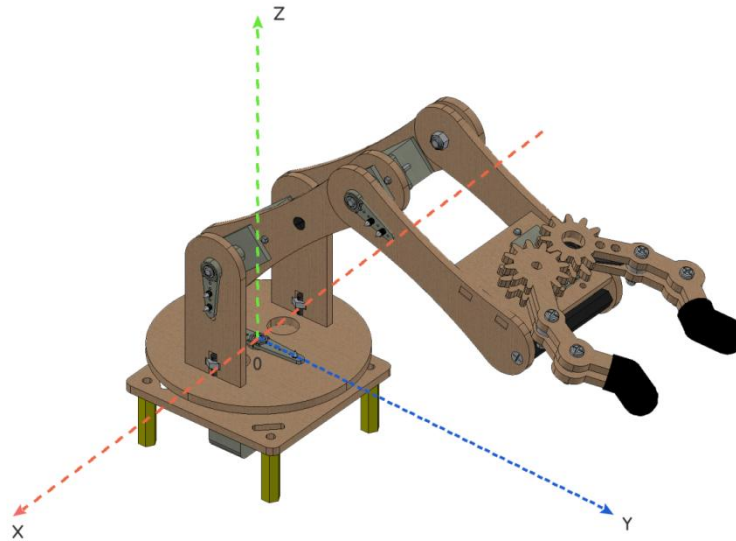
2. Three-Dimensional Cartesian Coordinate System

The three-dimensional Cartesian coordinate system consists of three perpendicular axes: the X-axis, Y-axis, and Z-axis. The Z-axis is perpendicular to the plane formed by the X-axis and Y-axis, representing the height in space. The point where all three axes intersect is called the origin, denoted as O. Any point in three-dimensional space can be represented by (x, y, z) , where x represents the point's vertical projection on the X-axis, y represents the point's vertical projection on the Y-axis, and z represents the point's vertical projection on the Z-axis.



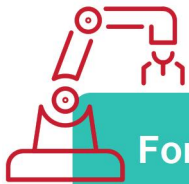
Since the space in which the robotic arm operates is three-dimensional, we use a three-dimensional Cartesian coordinate system to represent the position of the end effector in space. In this system, the origin of the coordinate axes is at the center of the base disk of the robotic arm, as shown in the figure below.





In the coordinate system above, the position of the robotic arm's end effector in the actual space is represented by (x, y, z) , with distance units in centimeters.

How does the robotic arm reach the corresponding spatial coordinate position?



Forward and Inverse Kinematics

Since the movement of the robotic arm's end effector is achieved by controlling the rotation angles of the joint servos, to make the robotic arm reach a specific point in the Cartesian coordinate system, it is necessary to determine the angles of each joint servo when the arm is at that position. There is a close correspondence between the spatial coordinates and the angles of the robotic arm's servos, and precise conversion between the two can be achieved through specific geometric calculations. This algorithm is known as the forward and inverse kinematics algorithm.



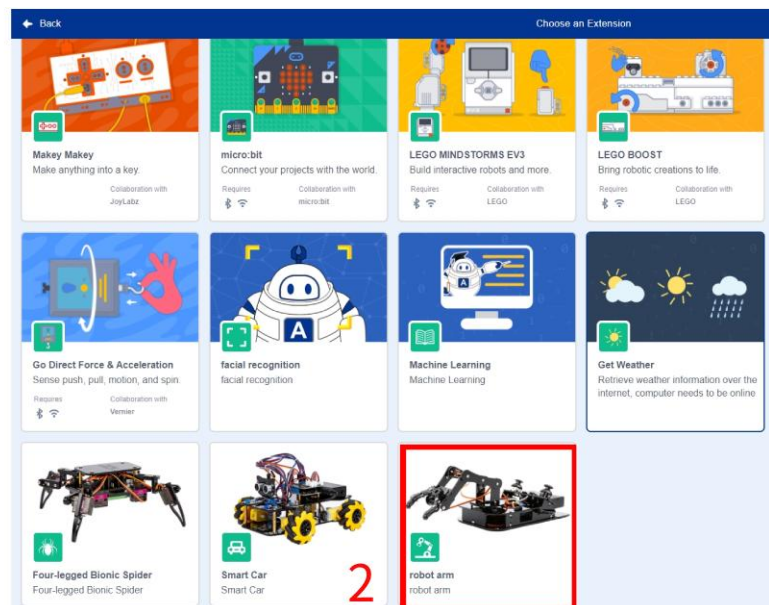
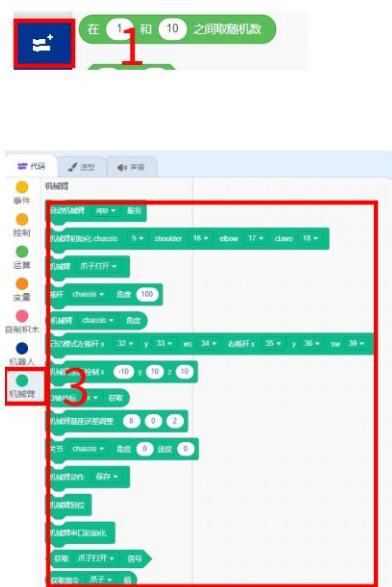


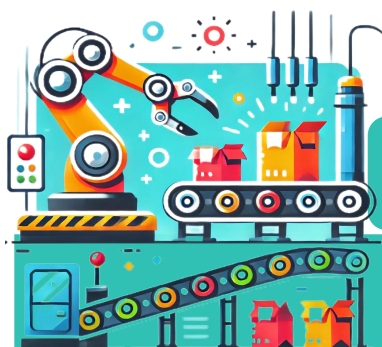
Forward Kinematics Algorithm: Given the angles of each joint servo, this algorithm calculates the position and orientation of the robotic arm's end effector in the Cartesian coordinate system.

Inverse Kinematics Algorithm: Given the coordinates of the robotic arm in the Cartesian coordinate system, this algorithm calculates the corresponding angles for each joint servo.

In this tutorial, we will introduce an extension for robotic arm control, which includes a variety of methods for controlling robotic arms. Among them, it includes spatial coordinate control methods for the robotic arm, eliminating the need for us to perform complex algorithmic transformations ourselves.

Click on "Add Extension" at the bottom left of ACECode, select the "Robotic Arm" extension, and you will then see that the "Robot Arm" category has been added to the instruction classification.





Smart Factory Lab

Coordinate Control Robotic Arm



Project Description

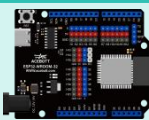
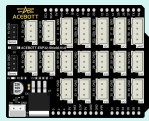


Control the robotic arm to automatically transport an object from point A to point B using spatial coordinates. To make controlling the robotic arm's movement using spatial coordinates more intuitive and convenient, this course will provide a coordinate map as a spatial reference.



Hardware Scheme

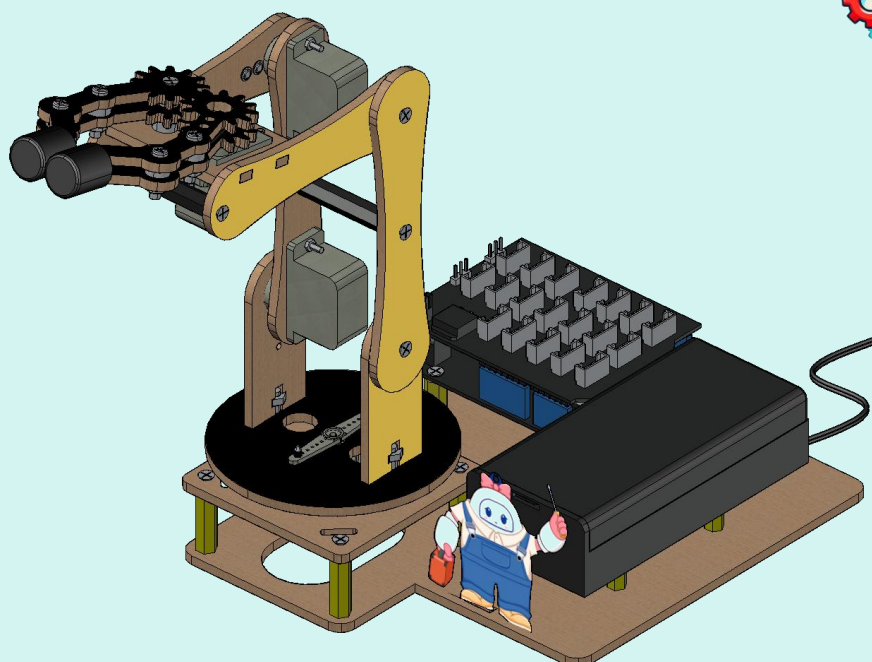
1. Hardware List

Picture	Name	Quantity
	ESP32 Board	1
	ESP32 Expansion Board	1
	MG90S Servo	4
	Type-C Data Cable	1



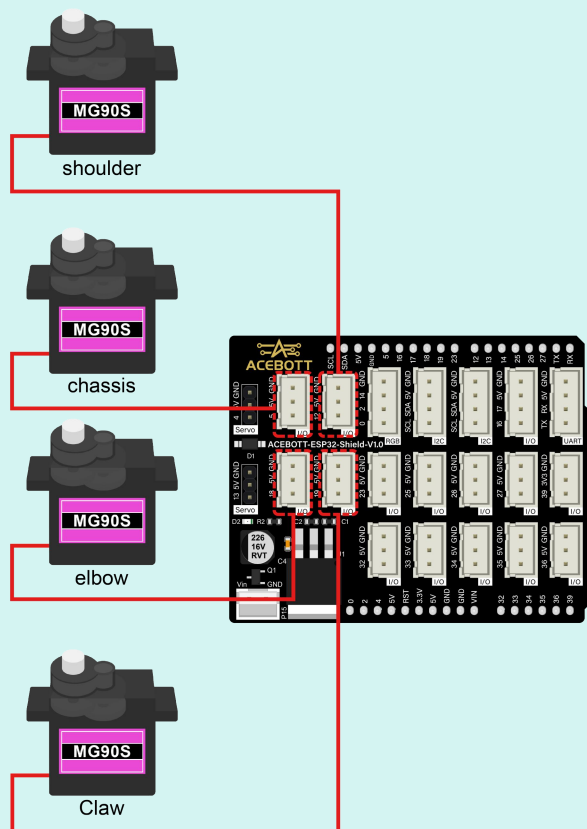


2. Hardware Structure



For detailed assembly steps, refer to Section 5 of the assembly document, titled "05 Coordinate Control Robotic Arm."

3. Hardware Wiring

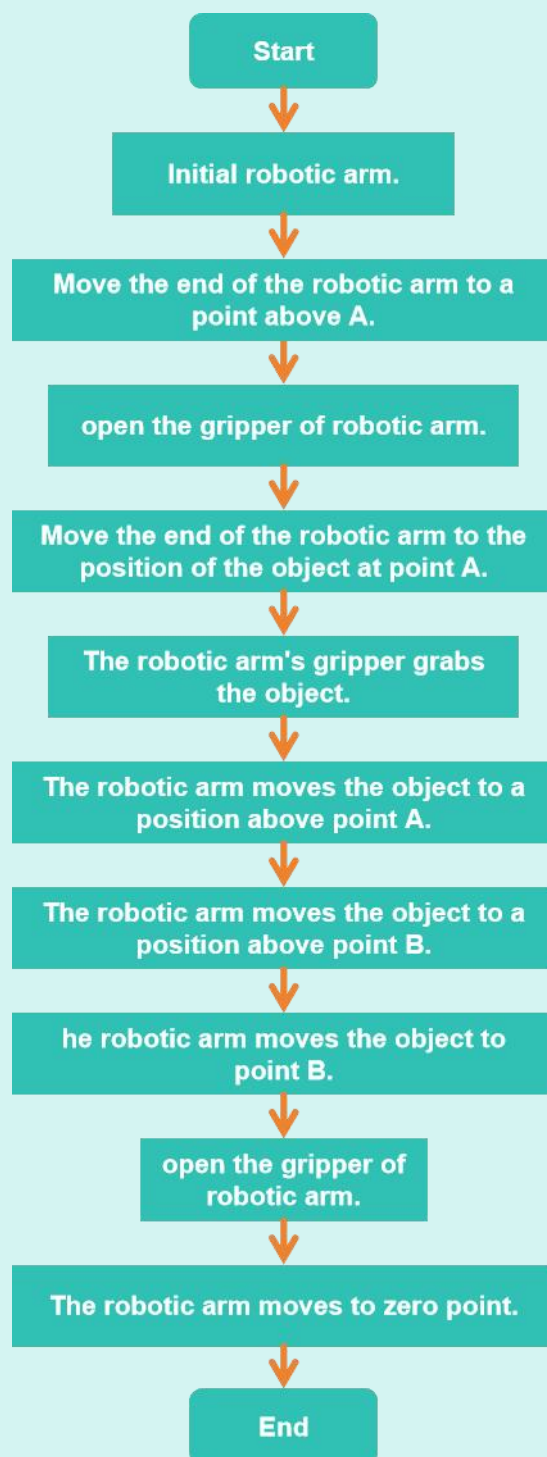




Programming



1. Program Flowchart





2. Coding

start the program

robot arm base error adjustment 5 0 2

robot arm initialization chassis 5 shoulder 12 elbow 18 claws 19

set A_x to -7

set A_y to 10

set A_z to 0

set B_x to 7

set B_y to 10

set B_z to 0

set Mid_z to 10

set claws angle 130

robot arm coordinate control x A_x y A_y z Mid_z

robot arm coordinate control x A_x y A_y z A_z

wait 1 seconds

set claws angle 90

wait 1 seconds

robot arm coordinate control x A_x y A_y z Mid_z

robot arm coordinate control x B_x y B_y z Mid_z

robot arm coordinate control x B_x y B_y z B_z

wait 1 seconds

set claws angle 130

wait 1 seconds

robot arm coordinate control x B_x y B_y z Mid_z

robot arm reset





3. Instruction Parsing

robot arm base error adjustment 8 0 2

: Due to the structure of the robotic arm

and the precision of the motors, there will be positional errors during the movement of the robotic arm. This command is used to adjust the angle error of the base, thereby making the spatial coordinate values more accurate.

The value in the first input box is used to adjust the central angle deviation of the Chassis servo. If it is to the left of the target position, the input value should be less than 0; on the contrary, the input value should be greater than 0. Following that are the second and third input boxes for fine-tuning. The second parameter is when the robotic arm is on the positive half-axis of the X-axis, if the end is still 1 degree to the right, write 1 directly; if it is 1 degree to the left, write -1; if there is no deviation, write 0. The third parameter is when the robotic arm is on the negative half-axis of the X-axis, if the end is still 1 degree to the right, write 1 directly; if it is 1 degree to the left, write -1; if there is no deviation, write 0.

robot arm initialization chassis 5 ▾ shoulder 16 ▾ elbow 17 ▾ claws 18 ▾

: This comma

nd is used for the initialization of the robotic arm and requires the input of the corresponding servo pin numbers.

set chassis ▾ angle 90

: This command is used to control the corresponding

joint of the robotic arm to rotate to a specified angle position. You can click the dropdown box to select different joints.





robot arm coordinate control x -10 y 10 z 10

: This command is used to move the end of the robotic arm to a specified coordinate position in space, where x is the X-axis coordinate value, y is the Y-axis coordinate value, and z is the Z-axis coordinate value.



Tips

During the grasping process, there may be certain errors due to the placement position or precision issues of the robotic arm.

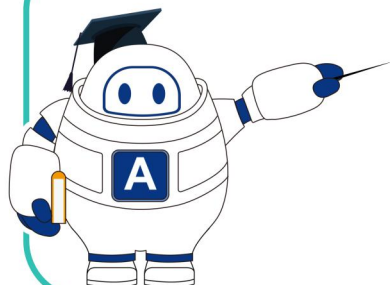


Section 6 Robotic Arm Palletizing



Course Objects

- Understand the role of palletizing.
- Comprehend the algorithm for robotic arm palletizing.
- Learn to implement intelligent palletizing with a robotic arm through programming.



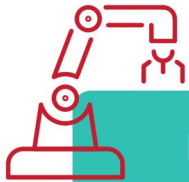
In a factory, how are goods stacked?

In a factory, to optimize the storage and transportation efficiency of goods, they are often carefully stacked according to specific sequences and rules. This method of storing goods is known as palletizing. In traditional factory settings, palletizing tasks are often performed manually, which is



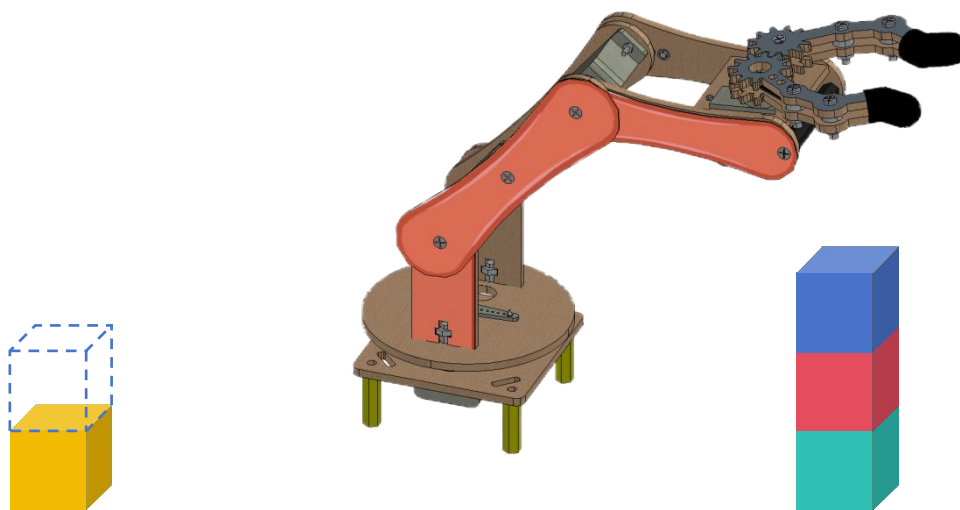


time-consuming, labor-intensive, and limited in efficiency. However, in smart factories, palletizing can now be automated by advanced robotic arms, significantly improving work efficiency and accuracy.



Principle of Robotic Palletizing

The basic principle of robotic palletizing is to programmatically control the robotic arm's movements to transport items from one location to a designated position and stack them according to a predetermined order and layering.



The main steps for robotic palletizing are:

1. Start: The end effector of the robotic arm moves to the initial position of the item.



2. Grasp: The end effector (such as a gripper) grabs the item, ensuring it is secure.

3. Transport: The robotic arm moves the grabbed item to the designated palletizing position.

4. Place: The robotic arm places the item according to the predetermined stacking order and layers.

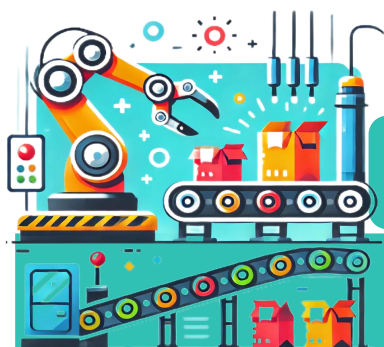
5. Repeat: The robotic arm repeats the above steps until all items are stacked.



Tips

When placing items, try to align the direction of the items with the direction of the robotic arm's end effector to facilitate grasping by the robotic arm.





Smart Factory Lab

Robotic Arm Palletizing



Project Description

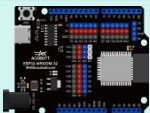
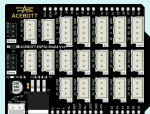


Program the robotic arm to accurately move the object from position A to position B and perform vertical stacking to ensure that the objects are securely arranged together.



Hardware Scheme

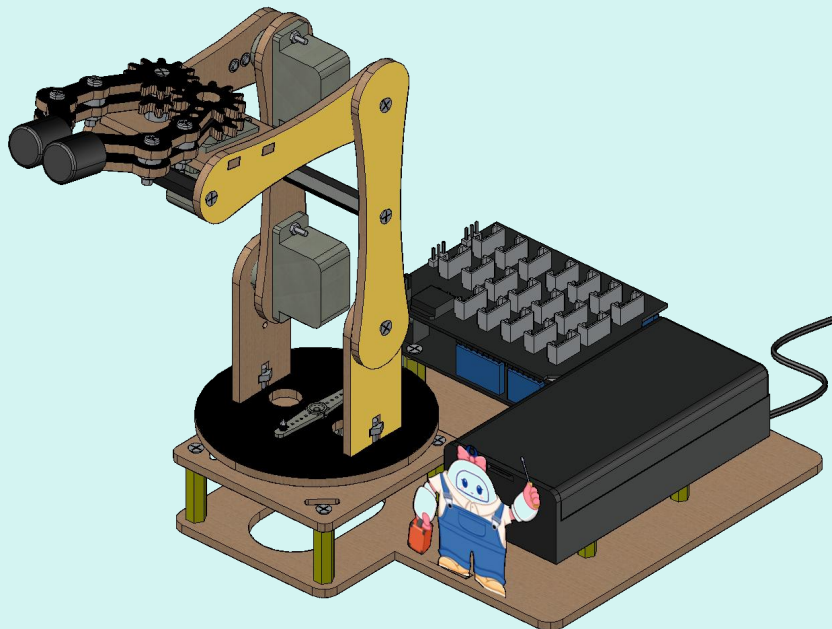
1. Hardware List

Picture	Name	Quantity
	ESP32 Board	1
	ESP32 Expansion Board	1
	MG90S Servo	4
	Type-C Data Cable	1



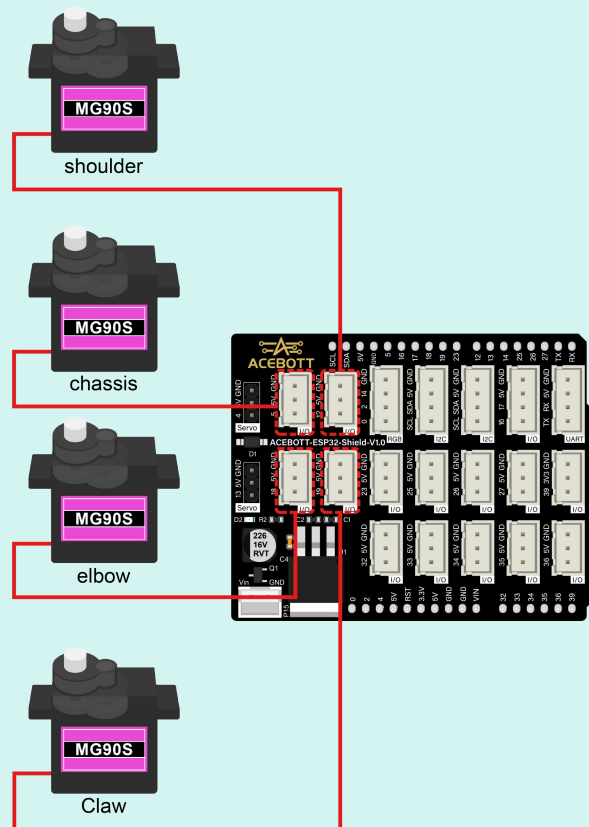


2. Hardware Structure



For detailed assembly steps, refer to Section 6 of the assembly document, titled "06 Robotic Arm palletizing."

3. Hardware Wiring

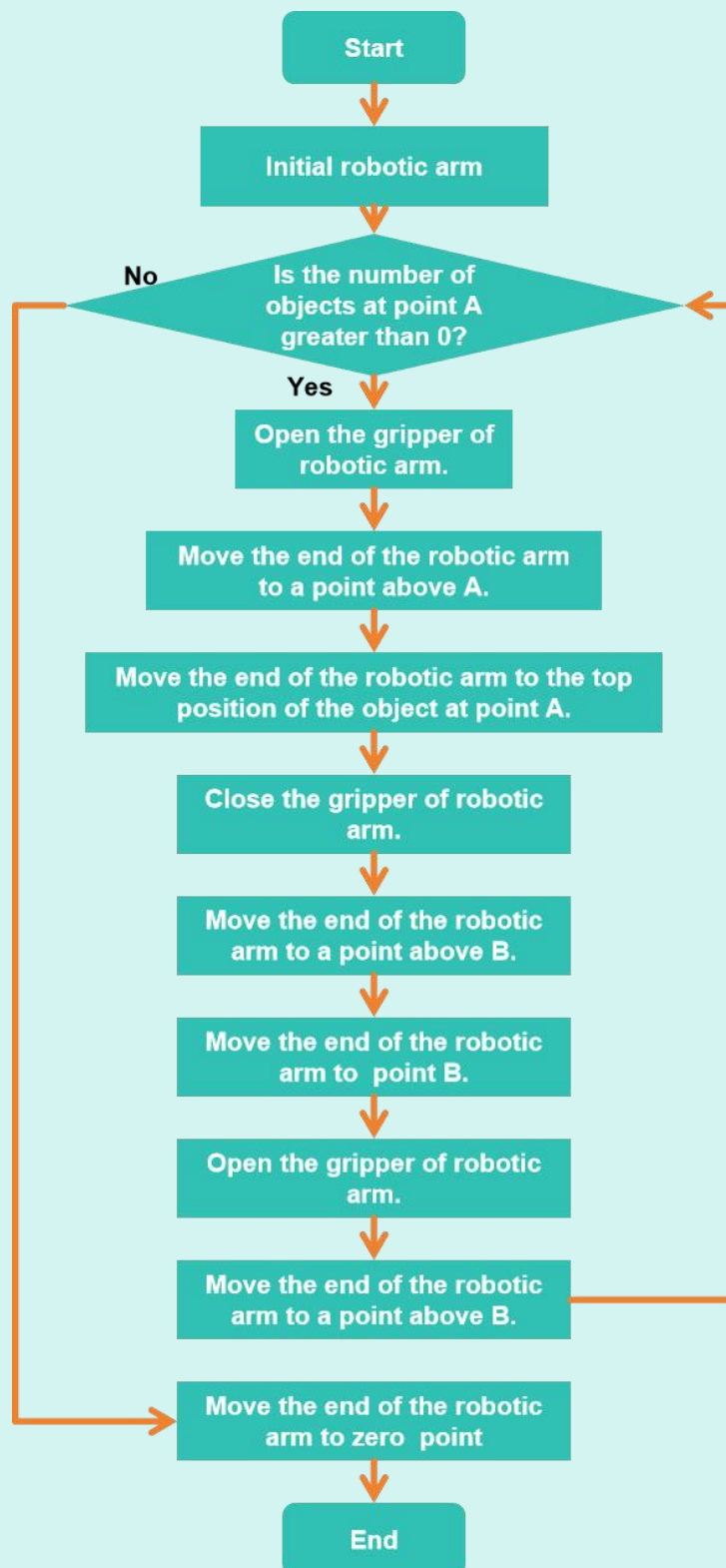




Programming

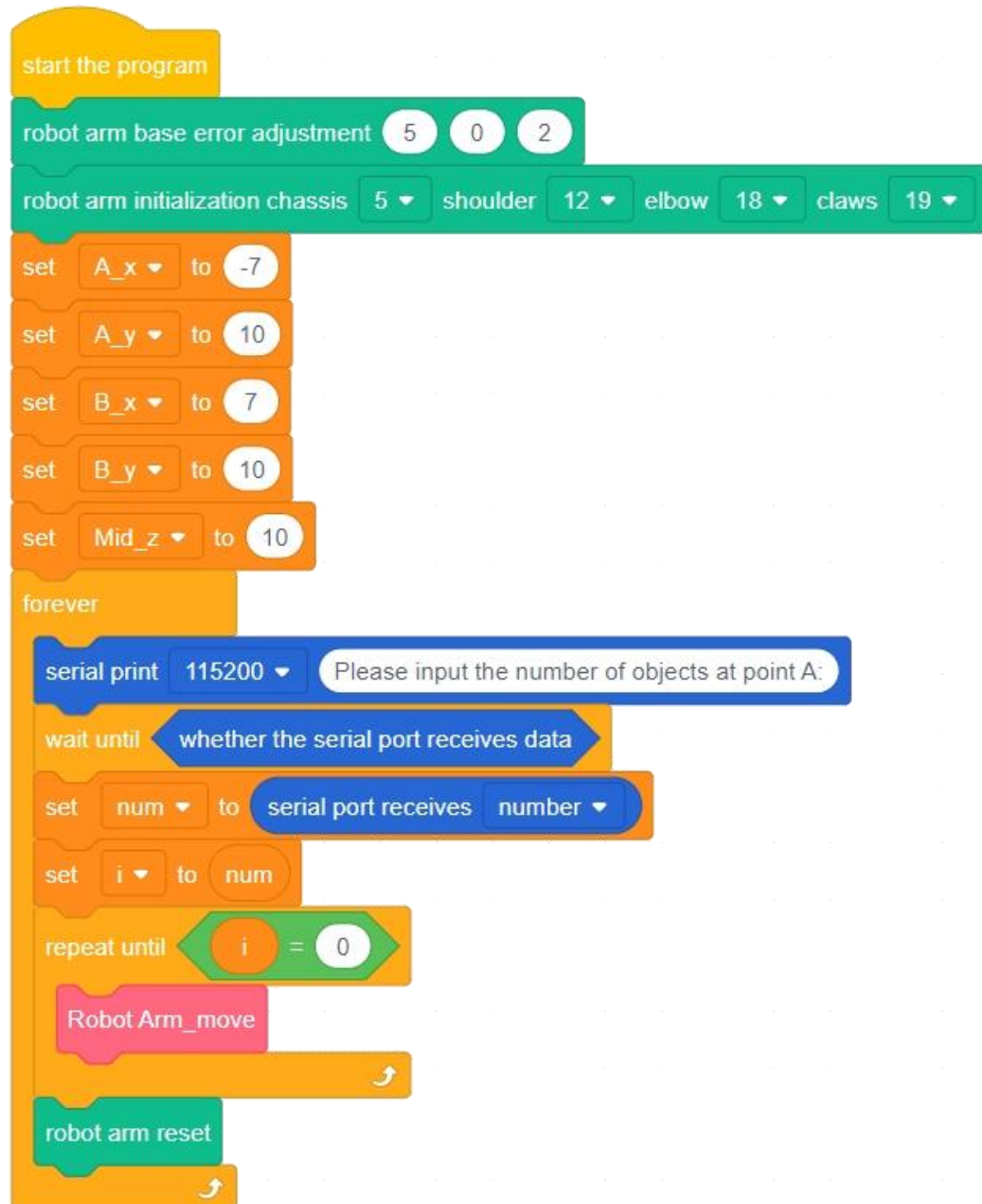


1. Program Flowchart





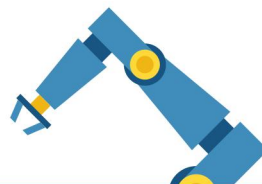
2. Coding





```
define Robot Arm_move
  set claws ▼ angle 130
  robot arm coordinate control x A_x y A_y z Mid_z
  robot arm coordinate control x A_x y A_y z i * 2 - 2
  wait 1 seconds
  set claws ▼ angle 90
  wait 1 seconds
  robot arm coordinate control x A_x y A_y z Mid_z
  robot arm coordinate control x B_x y B_y z Mid_z
  robot arm coordinate control x B_x y B_y z num - i * 2
  wait 1 seconds
  set claws ▼ angle 130
  wait 1 seconds
  robot arm coordinate control x B_x y B_y z Mid_z
  change i ▼ by -1
```



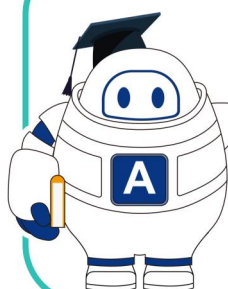


Section 7 Remote Control of Robotic Arm



Course Objects

- Understand the working principle of the joystick module.
- Master the use of the joystick module.
- Understand the concept of I2C.
- Implement robotic arm control through the joystick.



In more complex and variable environments, how can we enable the robotic arm to perform tasks more accurately?

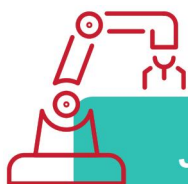
In complex, dynamic, or uncertain environments, autonomous robotic arms may struggle to adapt quickly to rapid changes. In such situations, human judgment becomes crucial for real-time control, enabling human-robot collaboration. Remote control of the robotic arm allows operators to make immediate





decisions and adjustments based on the current conditions, thereby avoiding operational mistakes caused by environmental changes or unexpected situations.

There are various ways to implement remote control of a robotic arm. In this lesson, we will use a joystick module to achieve remote control of the robotic arm.



Joystick Module

The joystick module is a common input device that consists of two potentiometers and a button. It is often used for controlling robots, game controllers, and similar devices.

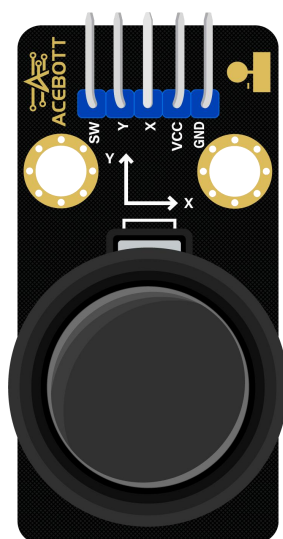
The two potentiometers in the joystick module are responsible for controlling the X-axis and Y-axis. When the joystick is moved along the X-axis, the resistance of the potentiometer on the X-axis changes, resulting in a change in the output voltage. Similarly, when the joystick is moved along the Y-axis, the resistance of the potentiometer on the Y-axis changes, causing a variation in the voltage signal.

When the joystick is pressed, the button in the joystick module closes, producing a low-level signal. If the joystick is not pressed, the output is a





high-level signal.



Joystick Module - Pin Analyse

SW: The button signal output, connects to the I/O pin of the controller board.

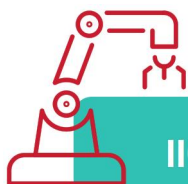
Y: Y-axis signal output pin, connects to the I/O pin of the controller board.

X: X-axis signal output pin, connects to the I/O pin of the controller board.

VCC: Connects to the 3.3v pin of the controller board.

GND: Connects to the GND pin of the controller board.

Due to the multiple pin resources required by the joystick module, we can use I2C communication to transmit data, thereby reducing the resource usage on the controller.



IIC Communication

IIC (Inter-Integrated Circuit), also known as I2C, is a commonly used interface communication protocol between hardware devices. Its design principle focuses on minimizing the number of signal lines while achieving high data rates. Fewer signal lines reduce pin usage, which was particularly important for early chips with limited pins.

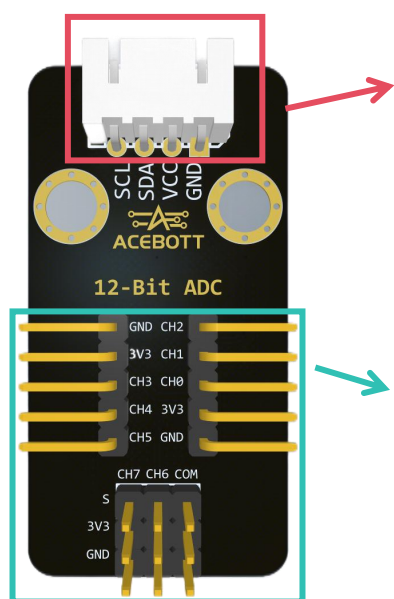




When using the IIC interface, you need to connect four wires: VCC, GND, SDA, and SCL. Among these, SDA and SCL occupy GPIO pins and can be connected to any corresponding IIC interface pins on the controller. SCL is the clock line that controls the timing, while SDA is the data line for transmitting data. In simple terms, IIC requires only two lines to transmit a large amount of data between multiple devices, reducing the number of I/O ports used on the controller.

In this lesson, we will use the ADS7828 chip to implement IIC communication between the joystick module and the ESP32 board. The ADS7828 chip is a single-supply, low-power 12-bit data acquisition device with 8 analog input ports and an IIC interface. It performs analog data sampling and transmits the sampled data to the control board via the IIC interface.

12 Bit ADC Module - Pin Analyse



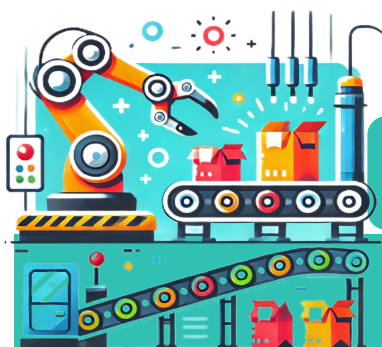
IIC Ports:

- SCL:** Connects to the SCL pin of the controller board.
- SDA:** Connects to the SDA pin of the controller board.
- VCC:** Connects to the VCC pin of the controller board.
- GND:** Connects to the GND pin of the controller board.

Data Input Ports:

- CH0 ~ CH7:** Analog input pin, converts analog signal to digital signal input.
- 3V3:** Power output pin, output 3.3V voltage.
- GND:** GND pin.





Smart Factory Lab

Remote Control Of the Robotic Arm



Project Description

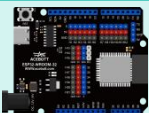
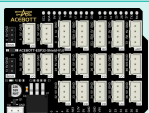
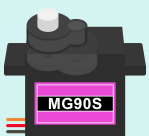
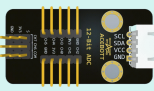


Create a remote controller using a joystick module, and use the controller to operate the movement of the robotic arm.



Hardware Scheme

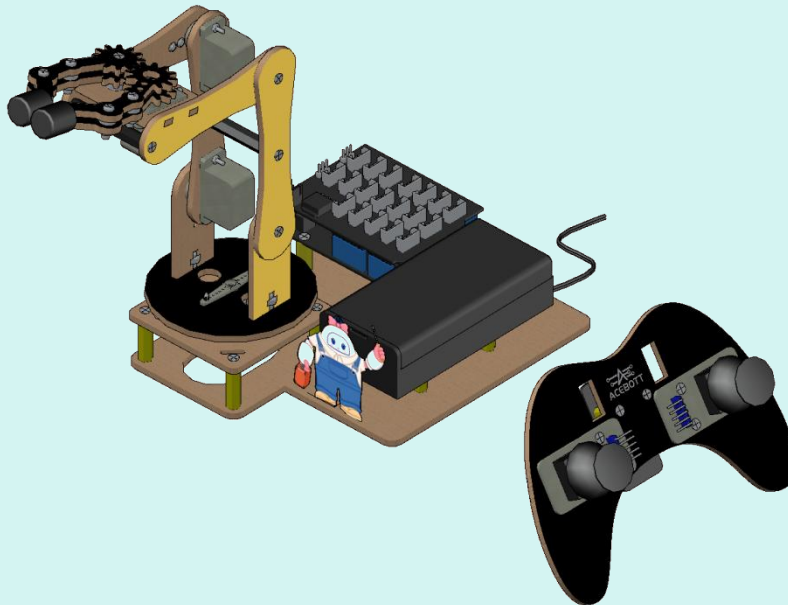
1. Hardware List

Picture	Name	Quantity
	ESP32 Board	1
	ESP32 Expansion Board	1
	MG90S Servo	4
	JoyStick Module	2
	12-Bit ADC Module	1
	Type-C Data Cable	1



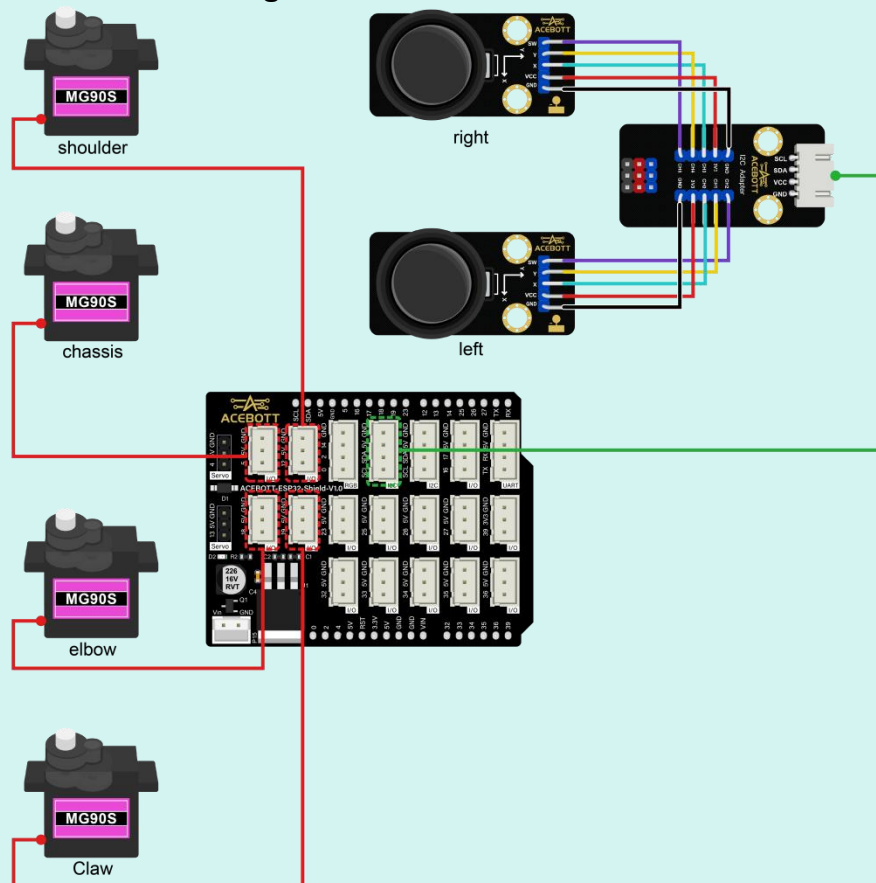


2. Hardware Structure



For detailed assembly steps, refer to Section 7 of the assembly document, titled "07 Remote Control of Robotic Arm."

3. Hardware Wiring

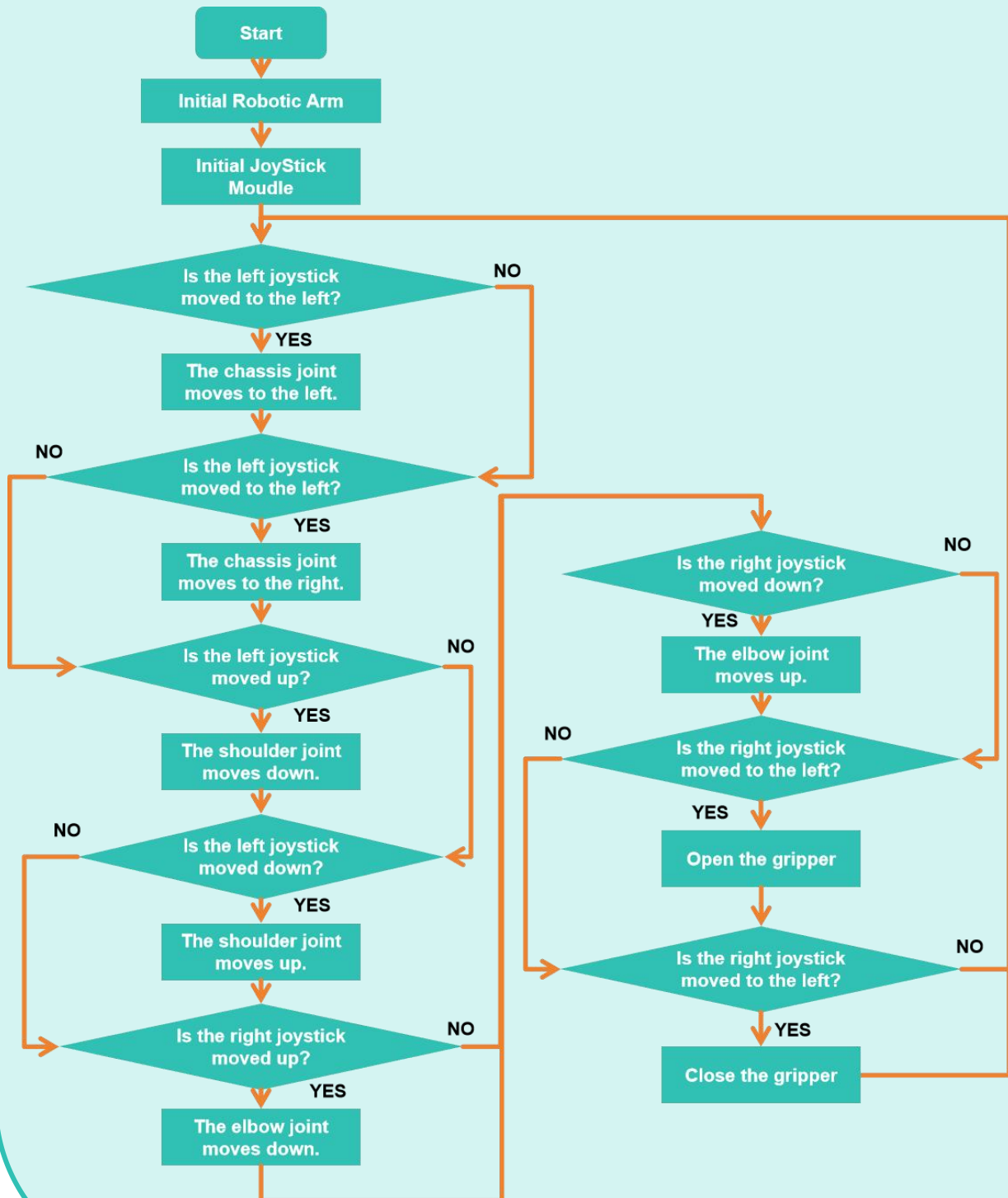




Programming



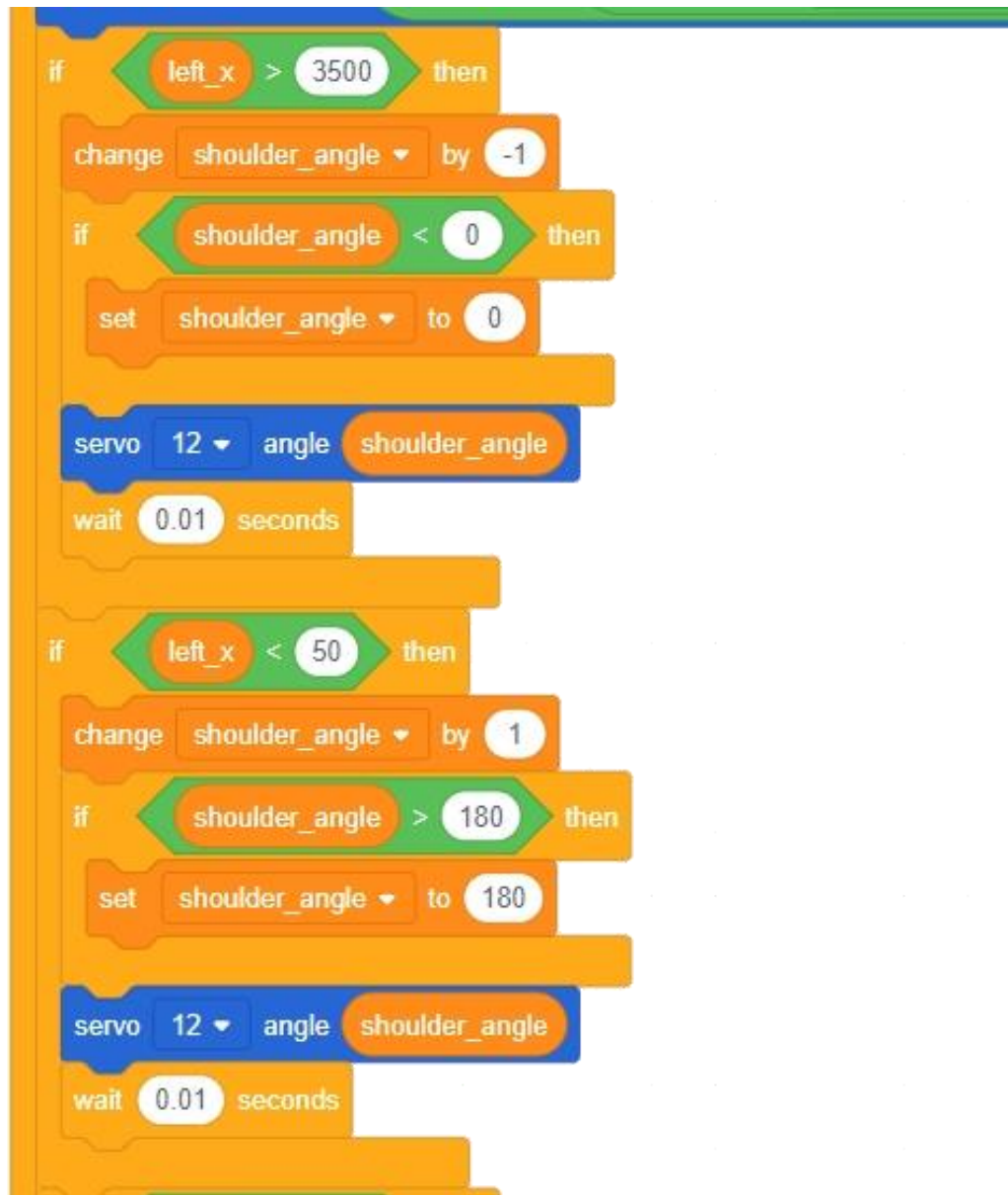
1. Program Flowchart

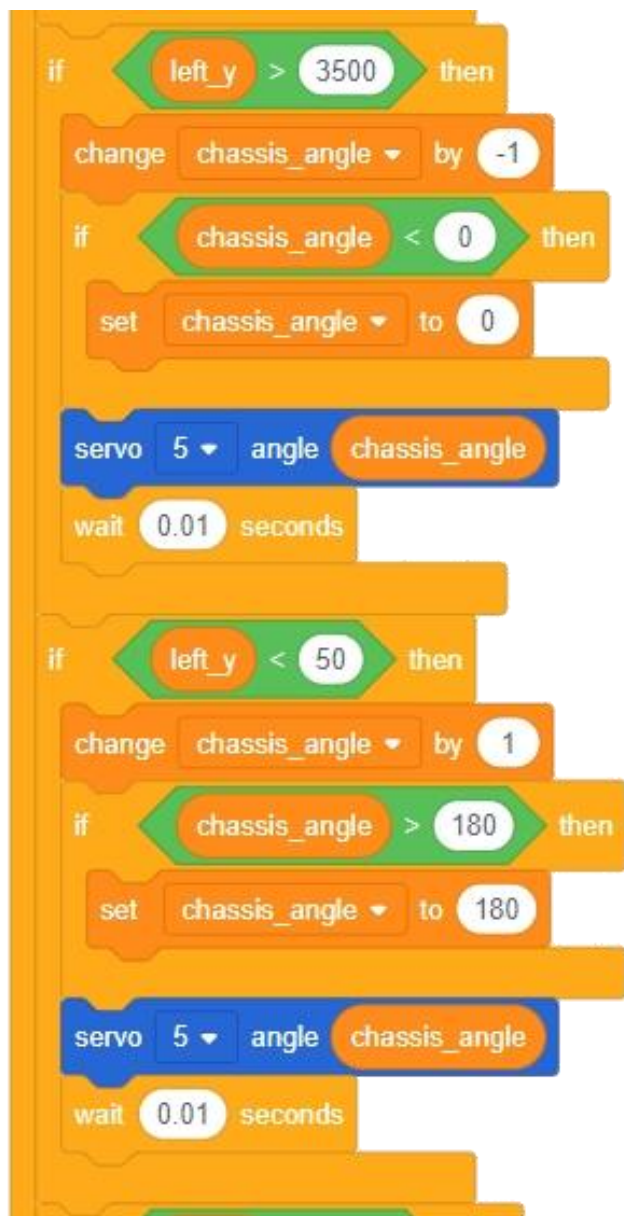


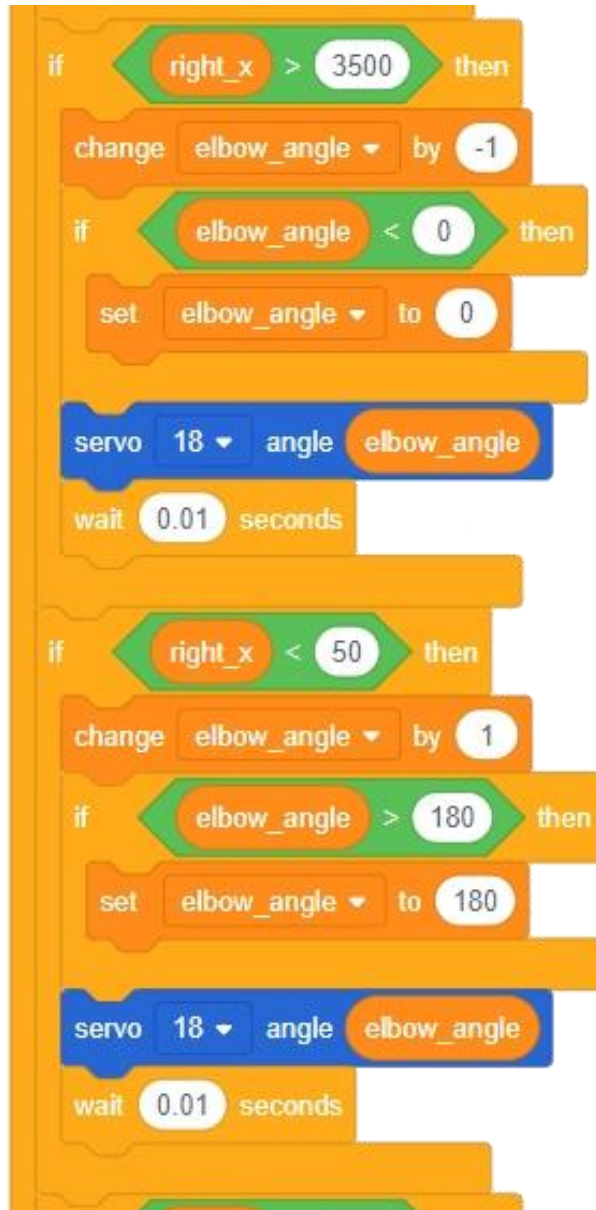


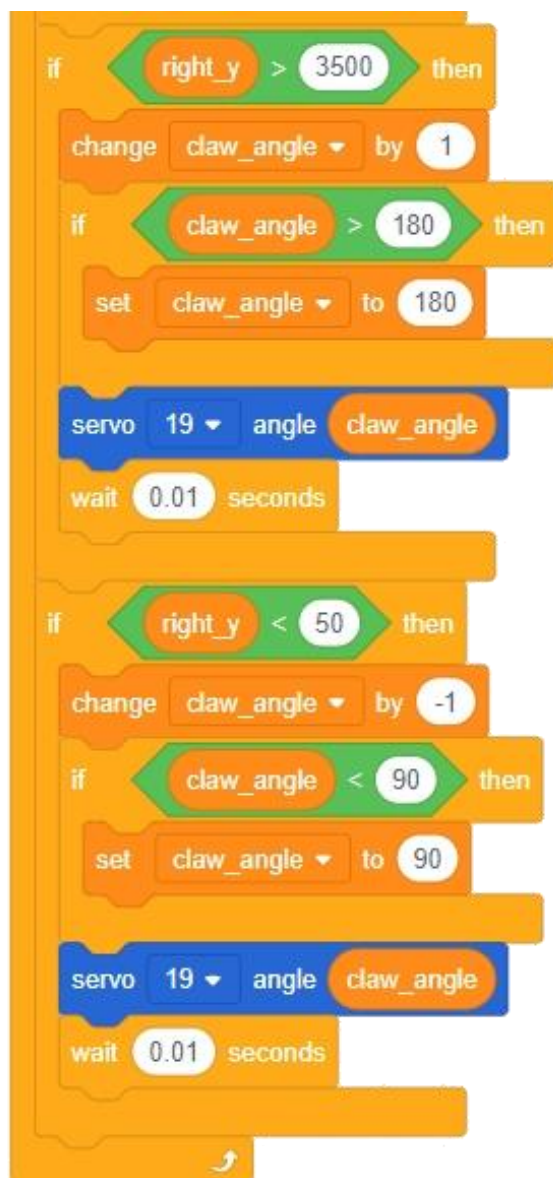
2. Coding













3. Instruction Parsing

left ▼ rocker x 0 ▼ y 1 ▼ sw 2 ▼ value x ▼

:This command is in the "Robots" instruction module, and it can obtain the value of the joystick module connected to the 12-bit ADC module.





Chapter 3 Smart Assembly Line

In factories, assembly lines are indispensable. They break down the entire production process into multiple steps, each managed by specialized workers. This reduces waiting times and unnecessary movement during production, thus enhancing efficiency. Within the assembly line operations, the conveyor system is the core equipment, capable of handling tasks such as material transport, product sorting, and quality inspection. In this chapter, we will explore the design and practical application of smart conveyor systems in intelligent assembly lines.



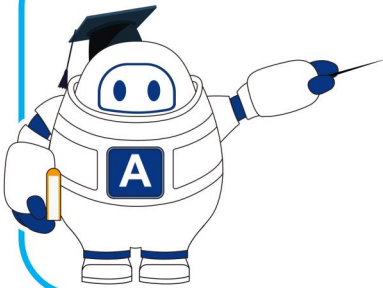


Section 1 Conveyor



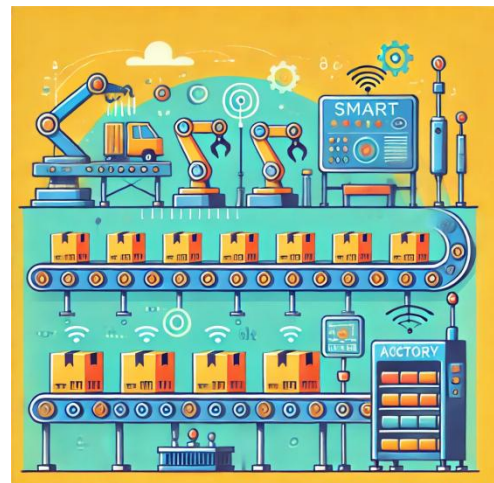
Course Objects

- Understand what an assembly line is.
- Understand the function of conveyor.
- Comprehend the structure and basic principles of conveyor belts.
- Understand the principles of DC motors.
- Learn how to use DC motors to drive conveyor belts.



What is an assembly line? How are products transported on an assembly line?

An assembly line is an industrial production method where the production process of a product is broken down into several consecutive and sequentially arranged operations. Each operation is typically carried out by specialized machinery or workers, and products sequentially pass through these operations to complete

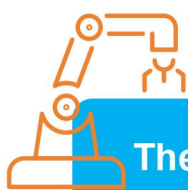




production. The main characteristics of an assembly line are its high degree of automation and efficiency, enabling large-scale production.

On an assembly line, products are primarily transported using a conveyor system. The conveyor moves products from one stage of production to the next, facilitating an automated and continuous production process.

How does a conveyor work?

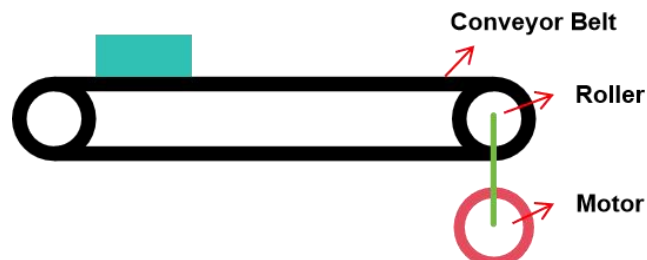


The Working Principle of Conveyor

The working principle of a conveyor is based on a continuous cycle motion driven by friction, primarily powered by an electric motor that drives the rollers allowing the conveyor belt to operate

on a fixed track, thus automating the transportation of materials or products.

The conveyor system mainly consists of the conveyor belt, transmission system, drive unit, tensioning device, and control system.



1. Conveyor Belt: The actual part that carries items, usually made of rubber, PVC, canvas, or other durable materials, providing sufficient durability and tensile strength, and mounted on rollers.





2. Transmission System: Typically consists of rollers. As the rollers rotate, the friction between them and the conveyor belt causes the conveyor belt to move in a cycle on a fixed track.

3. Drive Device: Generally composed of a motor, reduction gear, and coupling.

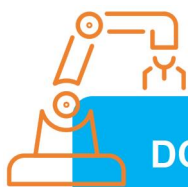
- **Motor:** The power source for the conveyor belt, driving the rollers to rotate.
- **Reduction Gear:** Converts the high-speed rotation of the motor into a low-speed, high-torque rotation suitable for conveyor belt operation.
- **Coupling:** Connects the motor and the rollers, transferring motor power to the rollers.

4. Tensioning: Device Used to adjust the tightness of the conveyor belt, ensuring that the conveyor belt does not slip due to being too loose or get damaged due to being too tight.

5. Control System: Composed of controllers and other devices, it outputs control commands for the movement of the conveyor belt.

The core of conveyor belt motion control is the control of the drive motor. In this lesson, we will learn how to use a DC reduction motor to drive the movement of the conveyor belt.

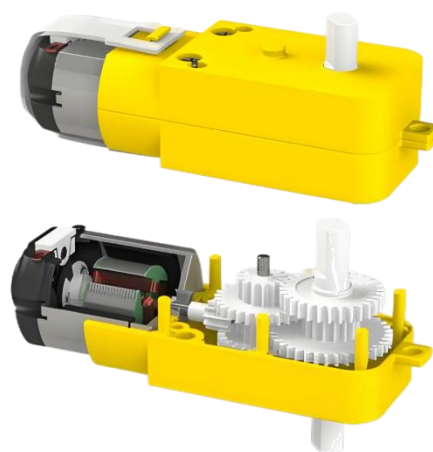




DC Reduction Motor

A DC reduction motor is a motor device that combines a DC motor with a reduction gear. It increases the motor's torque by reducing its speed, thus allowing the motor to handle larger loads. It is widely used in industrial automation, robotics, household appliances, and other fields.

The DC reduction motor used in this lesson is a TT motor, a small DC reduction motor with integrated reduction gears. The reduction gears provide a higher torque output, allowing the motor to drive heavier mechanical devices at low speeds. The operating voltage of the TT motor is 3V-6V, and its operating current is typically in the range of several hundred milliamperes. Under high load and stall conditions, the current can increase. The speed and direction of the motor can also be altered by changing the voltage input and the direction of the current.

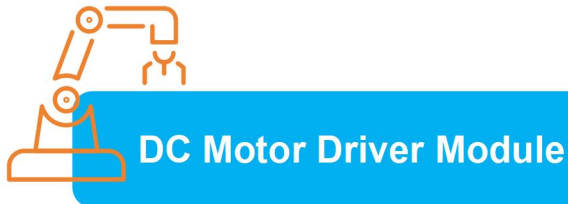


To ensure that the DC motor operates normally, sufficient voltage and current must be provided. However, a single pin on the ESP32 board can only output limited current (up to about 12mA), which is not enough to directly drive the





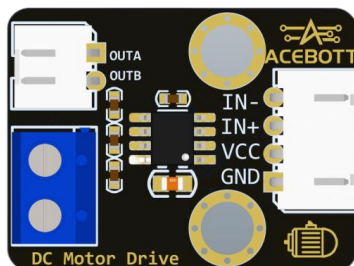
motor. Therefore, a DC motor driver module is needed to facilitate the normal operation of the motor.



A DC motor driver module is a circuit board specifically designed to control DC motors. It converts low-power control signals from microcontrollers (such as ESP32, Arduino, etc.) into high voltage and high current signals capable of driving DC motors. Driver boards typically include protective circuits to ensure safe and reliable control of motor forward and reverse rotation and speed.

In this lesson, we will use a DC motor driver module model HR1124S. This module is a single-channel H-bridge DC motor driver chip, capable of efficiently driving a single DC motor. When the IN- and IN+ pins receive different level signals, the motor starts running; when the IN- and IN+ pins receive the same level signal, the motor remains stationary.

DC Motor Driver Module - Pin Analyse



IN-: Signal input pin, connects to I/O pin of controller board.

IN+: Signal input pin, connects to I/O pin of controller board.

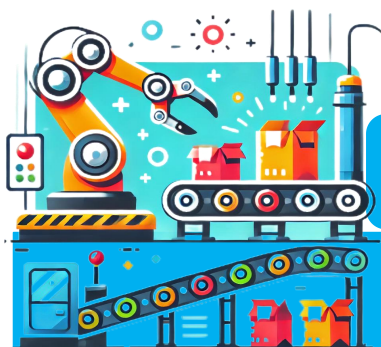
VCC: Connects to VCC pin of controller board.

GND: Connects to GND pin of controller board.

OUTA: Connects to pin of DC motor.

OUTB: Connects to pin of DC motor.





Smart Factory Lab

Conveyor Control



Project Description

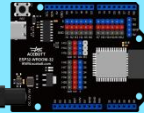
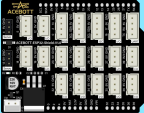


Follow the assembly steps in the building document to construct a conveyor belt and control it to start moving.



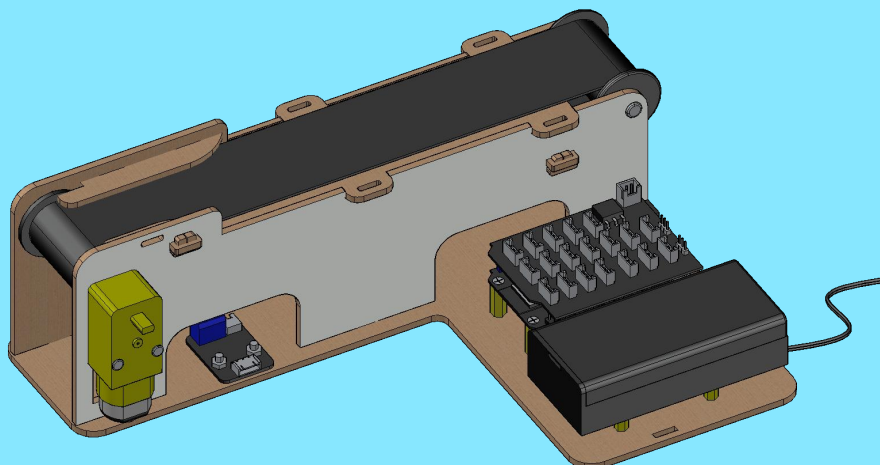
Hardware Scheme

1. Hardware List

Picture	Name	Quantity
	ESP32 Board	1
	ESP32 Expansion Board	1
	TT DC Motor	1
	DC Motor Driver Module	1
	Type-C Data Cable	1

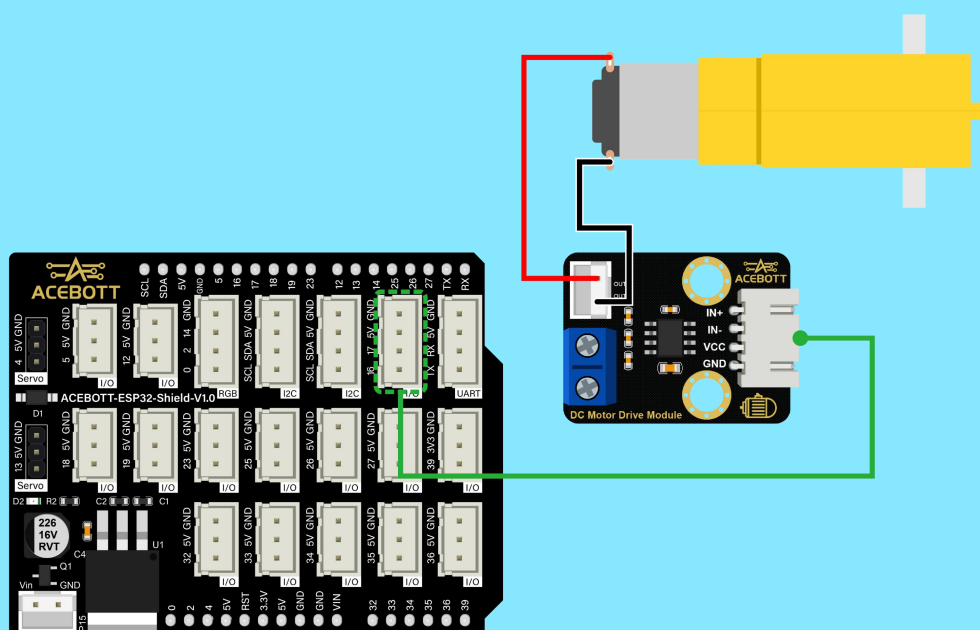


2. Hardware Structure



For detailed assembly steps, refer to Section 8 of the assembly document, titled "08 Conveyor Control."

3. Hardware Wiring

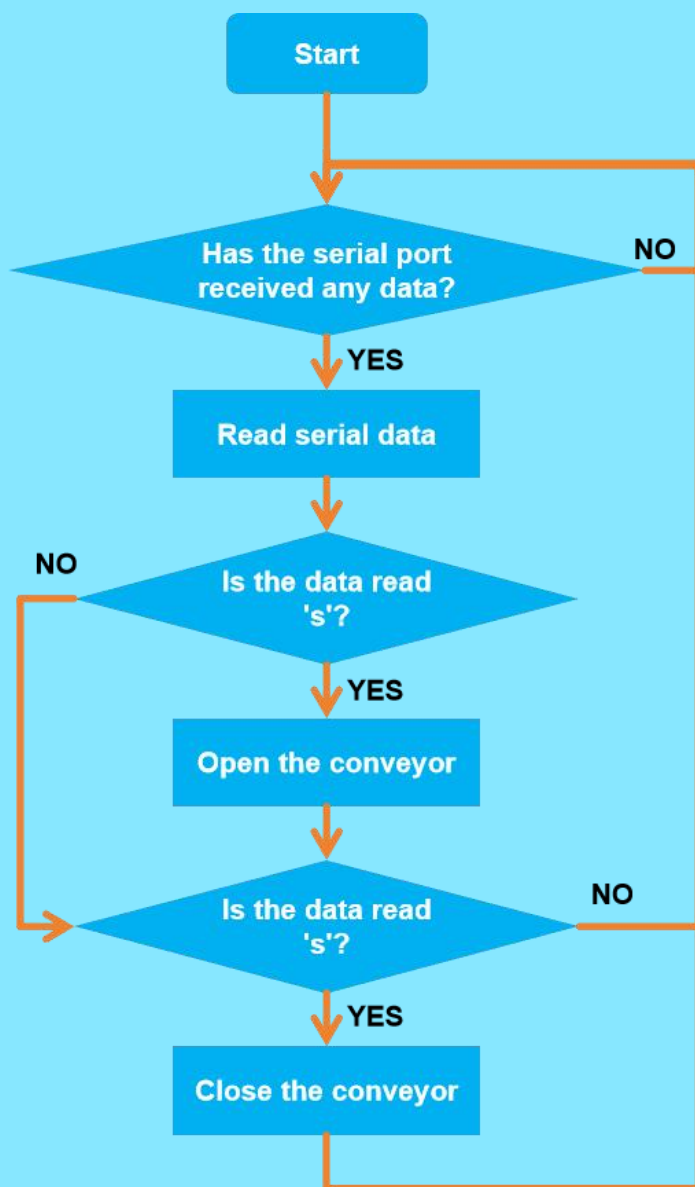




Programming

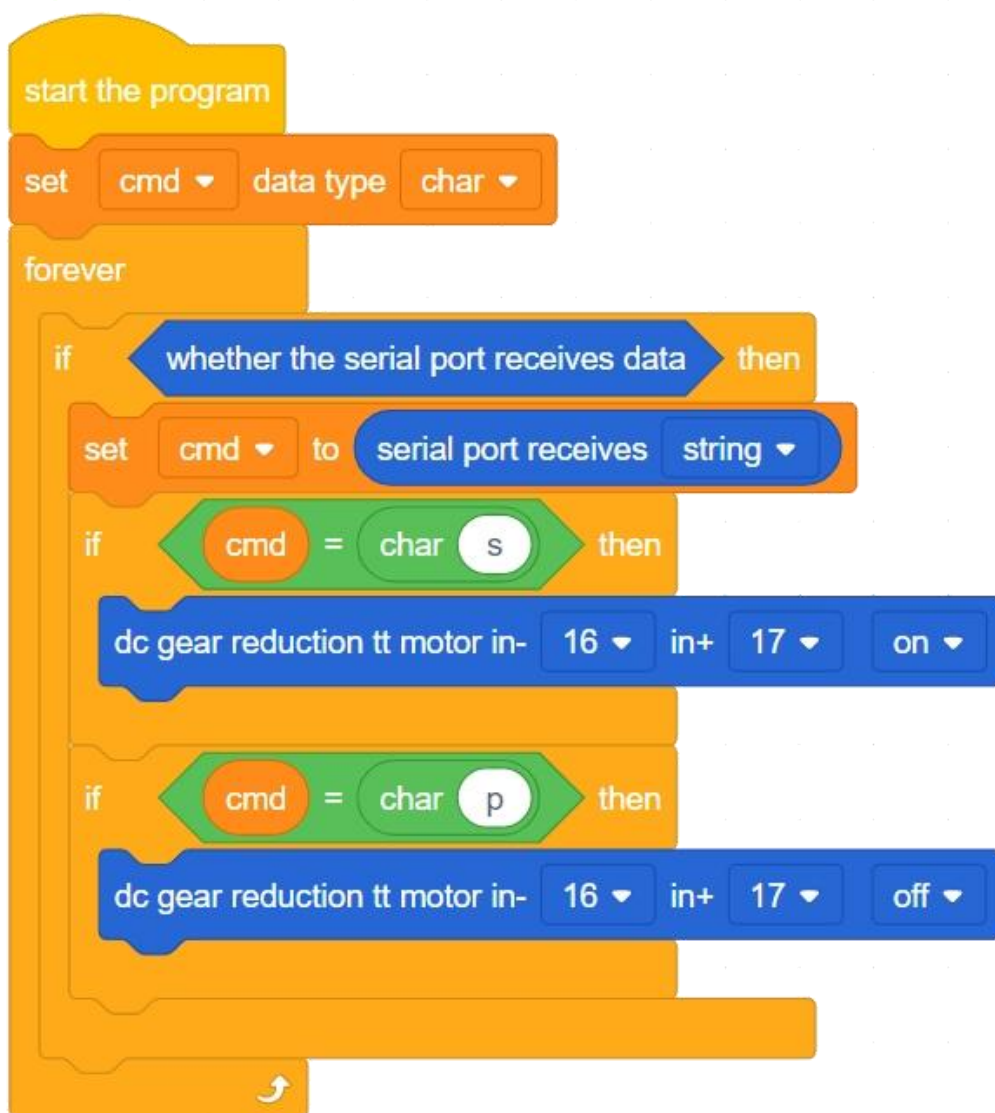


1. Program Flowchart





2. Coding





3. Instruction Parsing

dc gear reduction tt motor in- 13 ▼ in+ 13 ▼ on ▼

: This command is within the "Robots" instruction module. It is used to set the motor's on and off state, and the corresponding pin must be selected from the dropdown box in the command.

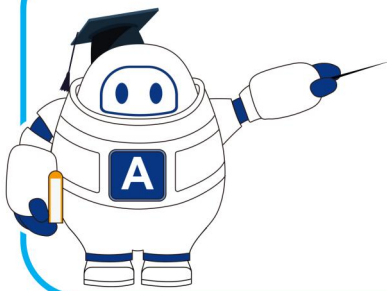




Section 2 Speed Control of Conveyor

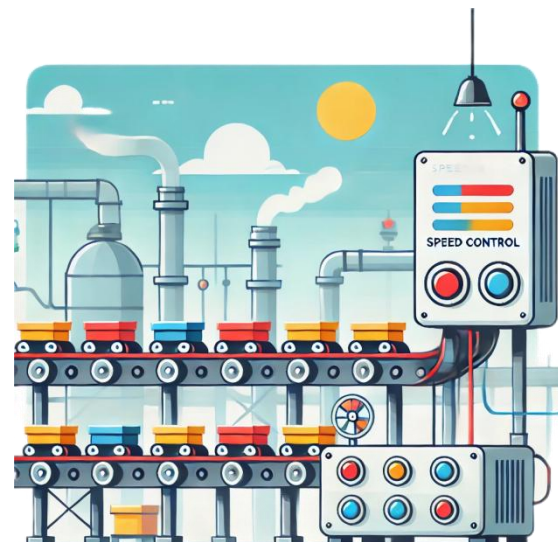
Course Objects

- Understand the concept of PWM (Pulse Width Modulation) signals.
- Learn how to output a PWM signal.
- Learn how to control the speed of a conveyor belt.



How to control the speed of conveyor?

In factory assembly line operations, we often need to adjust the speed of the conveyor belt because different stages of the production line require varying processing times. For example, the assembly, inspection, and packaging stages may each have different



requirements for conveyor speed. Additionally, in response to emergencies or temporary adjustment needs on the production line, adjusting the speed of the





conveyor belt allows us to quickly respond to new demands, thus preventing production line stalls or backlog of items.

So, how do we control the speed of the conveyor belt?

Adjusting the speed of the conveyor belt essentially involves controlling the speed of a DC motor. As we learned in the previous lesson, the speed of the motor can be controlled by adjusting the input voltage. Within the normal operating voltage range of the motor, a lower voltage slows down the motor speed, while a higher voltage increases the speed.

For the ESP32 board, the motor speed can be controlled by outputting different voltages through PWM signals.



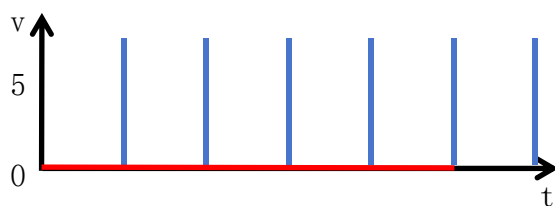
PWM (Pulse Width Modulation) is a digital signal modulation technique used to control analog circuits. It modulates different voltages by adjusting the width of pulse signals (i.e., the duty cycle) to control the operation of devices, such as motor speed and LED brightness. A PWM signal is a digital signal that rapidly switches between a high voltage (usually V_{cc} , such as 5V or 3.3V) and a low voltage (0V) within a fixed cycle. The duration of the high voltage within each cycle is called the pulse width. The duty cycle refers to the ratio of the high





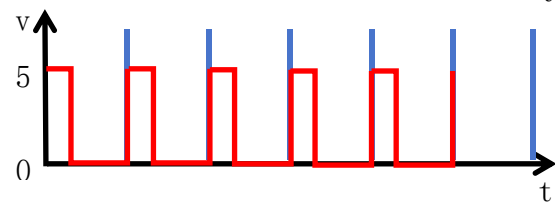
voltage duration to the total cycle time. The duty cycle can vary from 0% (signal always at low level) to 100% (signal always at high level). By changing the duty cycle, the average voltage of the output signal can be controlled.

In ACECode programming,  instruction can be used to output a PWM signal to a TT motor, where the first two dropdown boxes are for entering the pins that are connected; the input box is for entering the duty cycle of the PWM signal output by the second pin, with a value range of 0 to 255, corresponding to a duty cycle of 0% to 100%.



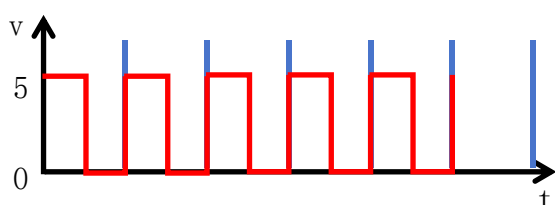
dc gear reduction tt motor in- 13 in+ 13 0

0% Duty Cycle--0V



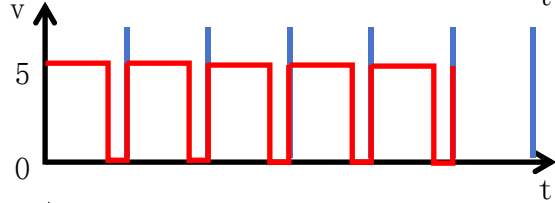
dc gear reduction tt motor in- 13 in+ 13 64

25% Duty Cycle--1.25V



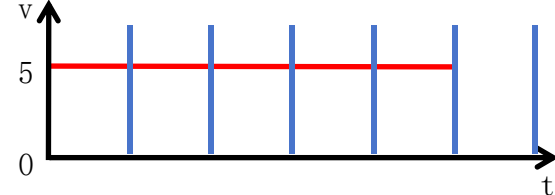
dc gear reduction tt motor in- 13 in+ 13 127

50% Duty Cycle--2.5V



dc gear reduction tt motor in- 13 in+ 13 191

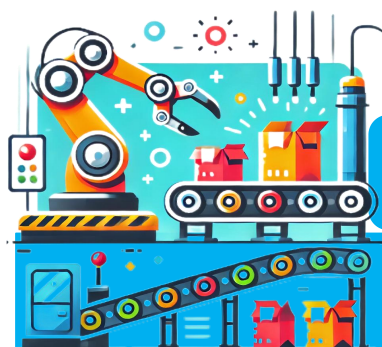
75% Duty Cycle--3.75V



dc gear reduction tt motor in- 13 in+ 13 255

100% Duty Cycle--5V





Smart Factory Lab

Speed Control of Conveyor



Project Description

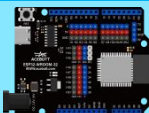
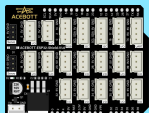


Control the speed of the conveyor belt using keyboard keys, and divide the conveyor belt speed into three levels.



Hardware Scheme

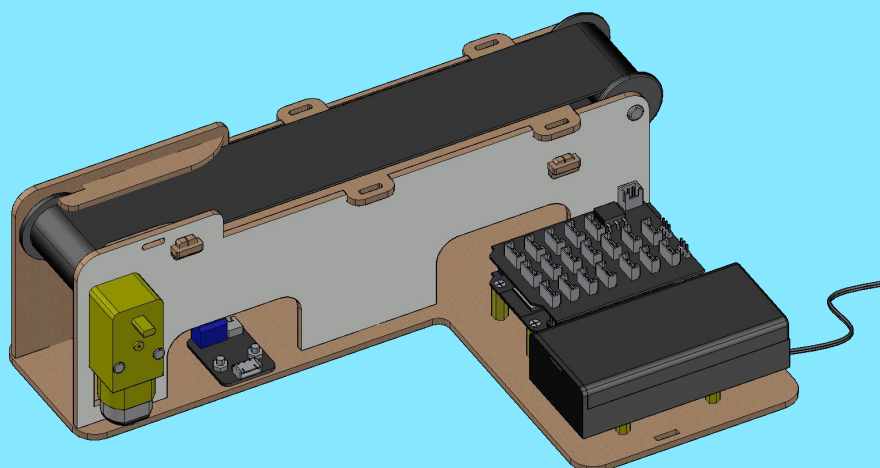
1. Hardware List

Picture	Name	Quantity
	ESP32 Board	1
	ESP32 Expansion Board	1
	TT DC Motor	1
	DC Motor Driver Module	1
	Type-C Data Cable	1



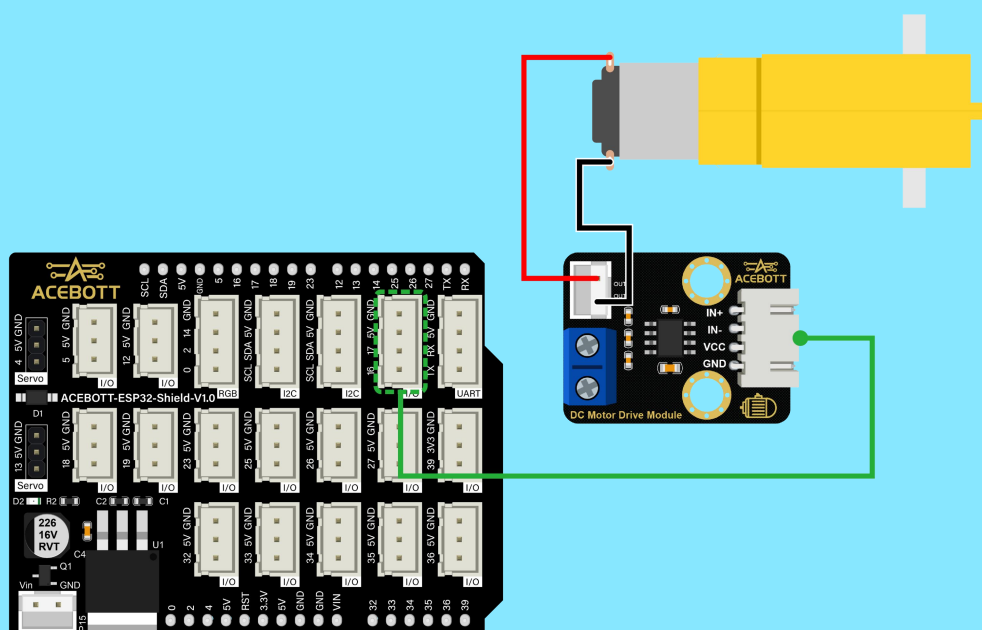


2. Hardware Structure



For detailed assembly steps, refer to Section 9 of the assembly document, titled "09 Speed Control of Conveyor."

3. Hardware Wiring

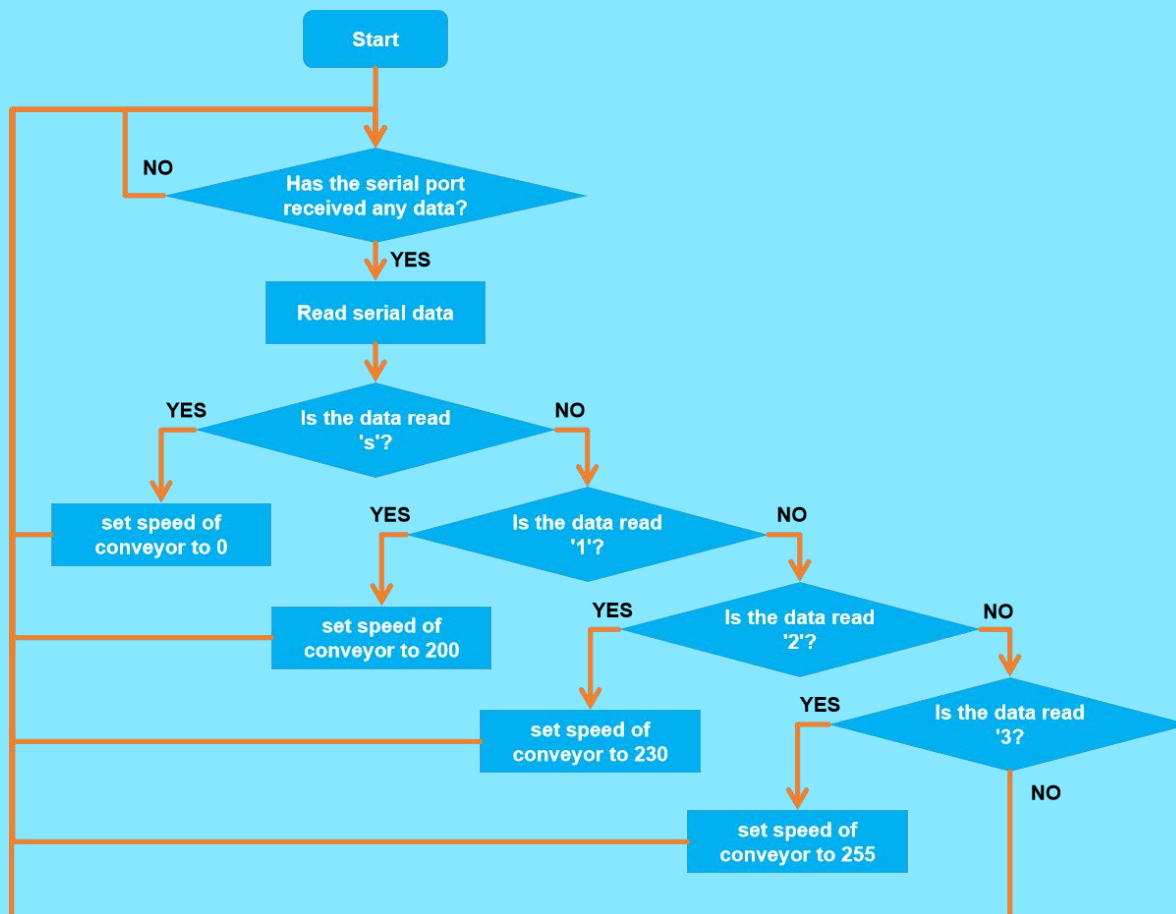




Programming

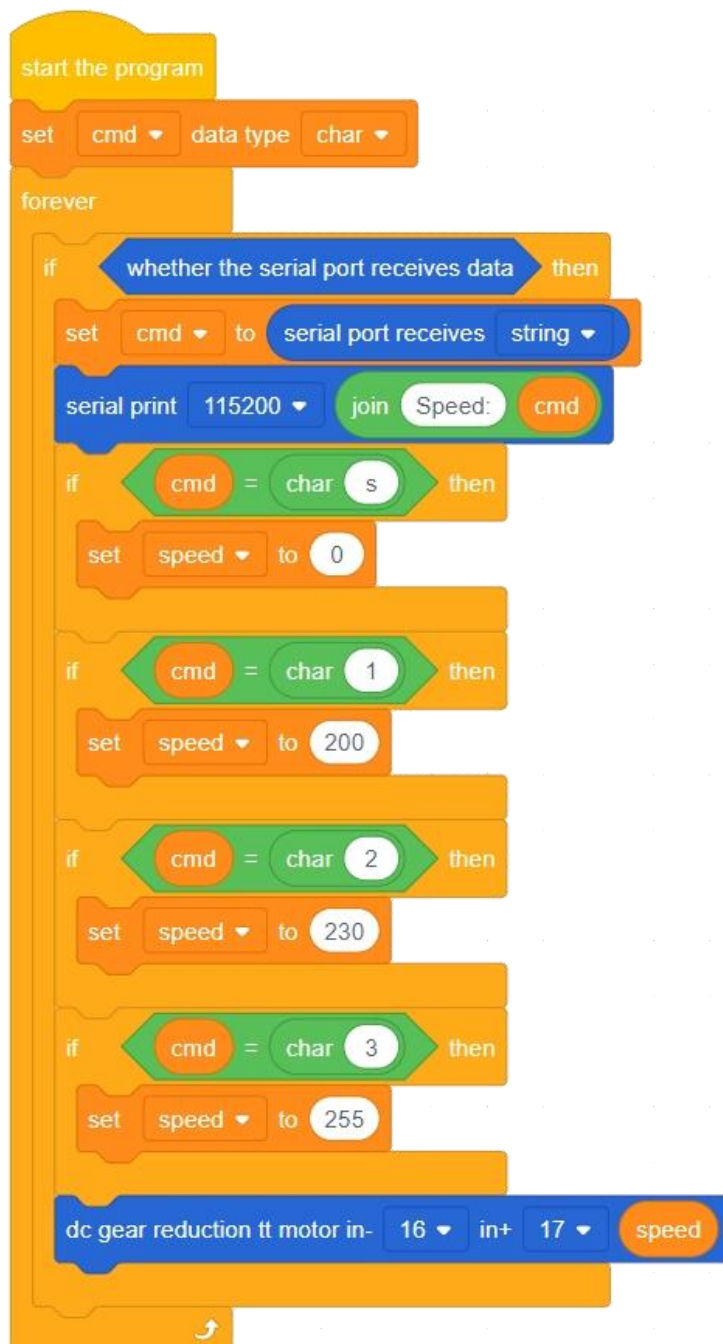


1. Program Flowchart





2. Coding



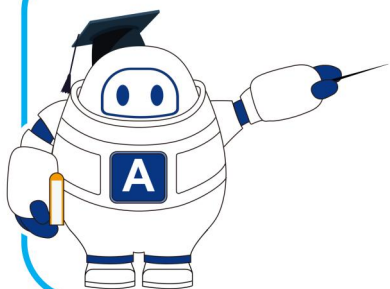


Section 3 Smart Conveyor



Course Objects

- Understand infrared obstacle avoidance sensor.
- Master the use of infrared obstacle avoidance sensor.
- Build a smart conveyor using infrared obstacle avoidance sensor.



During the operation of the conveyor , how can it be made to automatically stop running if there are no goods on it?

Our previously designed conveyor belt continues to operate even when there are no items on it, leading to several major issues:

1. Energy Waste: When the conveyor belt runs empty, the electric motor and other drive devices continue to consume electricity. This unnecessary energy consumption increases the operational costs of the factory.

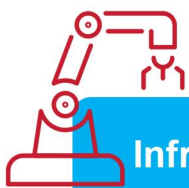




2. Equipment Wear: The mechanical components of the conveyor belt, including the motor, rollers, and belt, endure friction and stress during operation. Continuous operation without items can lead to excessive wear, thereby shortening the lifespan of the equipment.

3. Noise Pollution: The conveyor belt generates mechanical noise during operation. Continuous operation without items can negatively impact the factory's working environment due to persistent noise.

Given these issues, we hope the conveyor belt can automatically stop when no objects are present and automatically start when objects are detected. In this lesson, we will use infrared obstacle avoidance sensors to build a smart conveyor belt that meets these requirements.

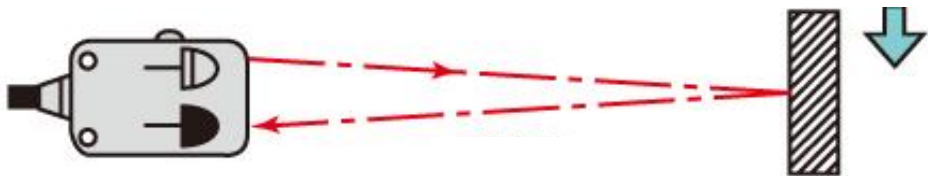


Infrared Obstacle Avoidance Sensor

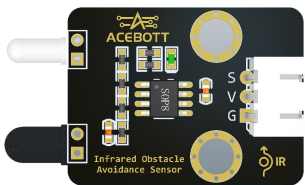
The infrared obstacle avoidance sensor is a common electronic sensor used to detect the presence of obstacles. It typically consists of an infrared emitter and an infrared receiver. The working principle of the infrared obstacle avoidance sensor is based on the reflection of infrared light. When the infrared emitter emits a beam of infrared light, if there is an object ahead, the infrared light is



reflected back by the surface of the object. The infrared receiver then captures the reflected light and converts the light signal into an electrical signal for output.

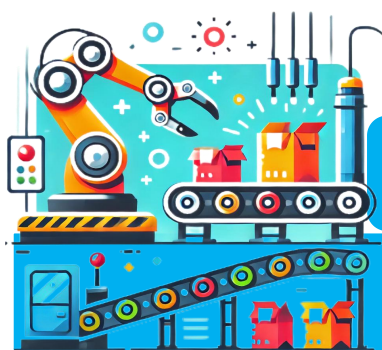


The infrared obstacle avoidance sensor used in this lesson has a detection range of about 10cm. When an object is detected, the signal pin outputs a low level; when no object is detected, the signal pin outputs a high level.



Infrared Obstacle Avoidance Sensor - Pin Analyse

- S:** Signal output pin, connects to the I/O pin of controller board.
- V:** Connects to the 5V pin of controller board.
- G:** Connects to the GND pin of controller board.



Smart Factory Lab

Smart Conveyor



Project Description

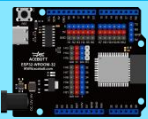
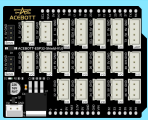


Build a smart conveyor belt that automatically operates when there are objects on it and automatically stops when there are no objects.



Hardware Scheme

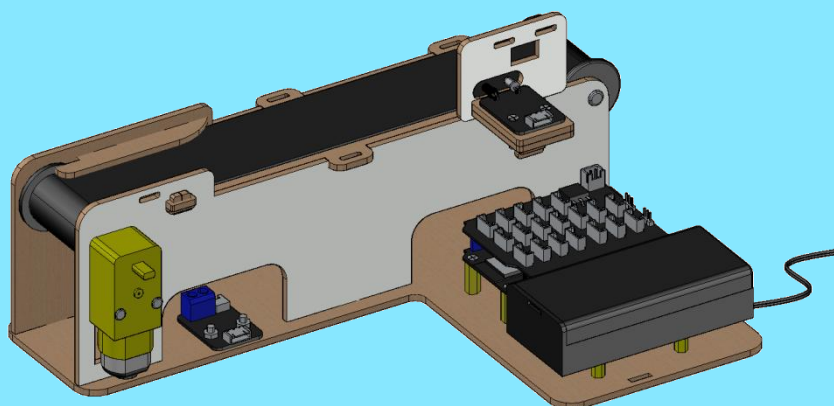
1. Hardware List

Picture	Name	Quantity
	ESP32 Board	1
	ESP32 Expansion Board	1
	TT DC Motor	1
	DC Motor Driver Module	1
	Infrared Obstacle Avoidance Sensor	1
	Type-C Data Cable	1



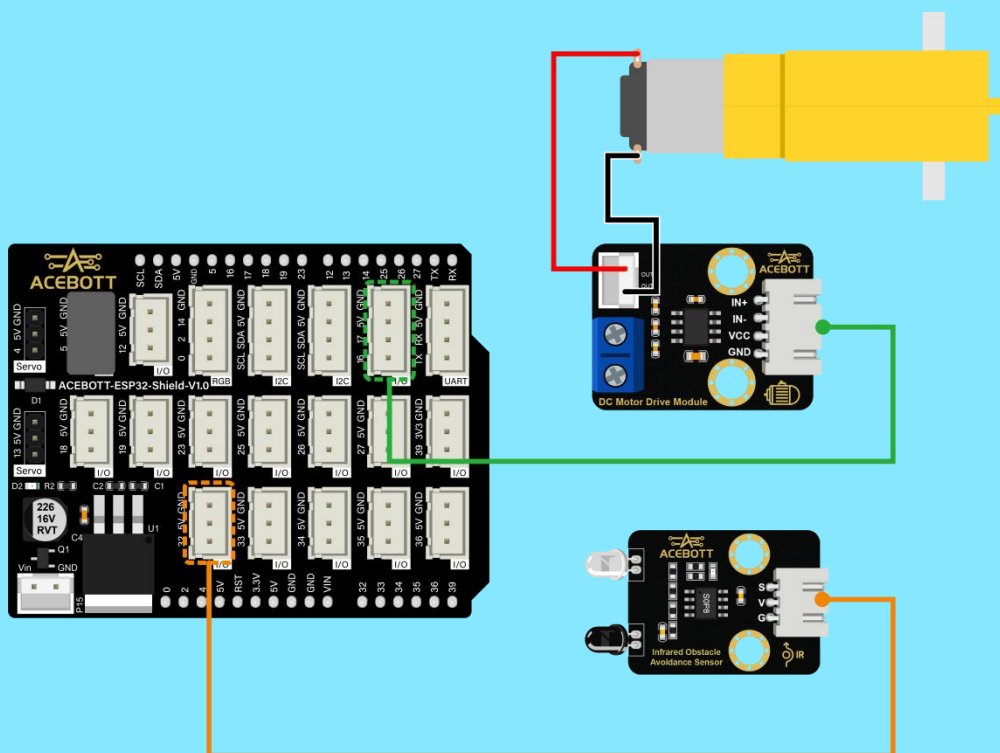


2. Hardware Structure



For detailed assembly steps, refer to Section 10 of the assembly document, titled "10 Smart Conveyor".

3. Hardware Wiring

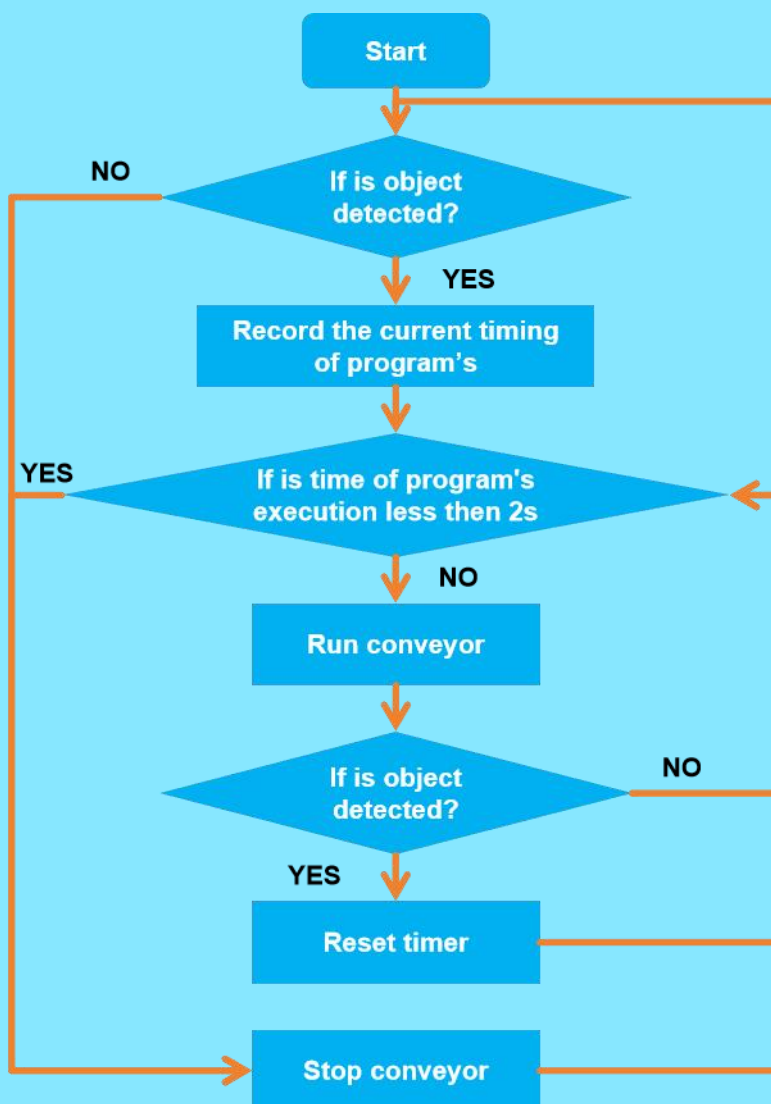




Programming

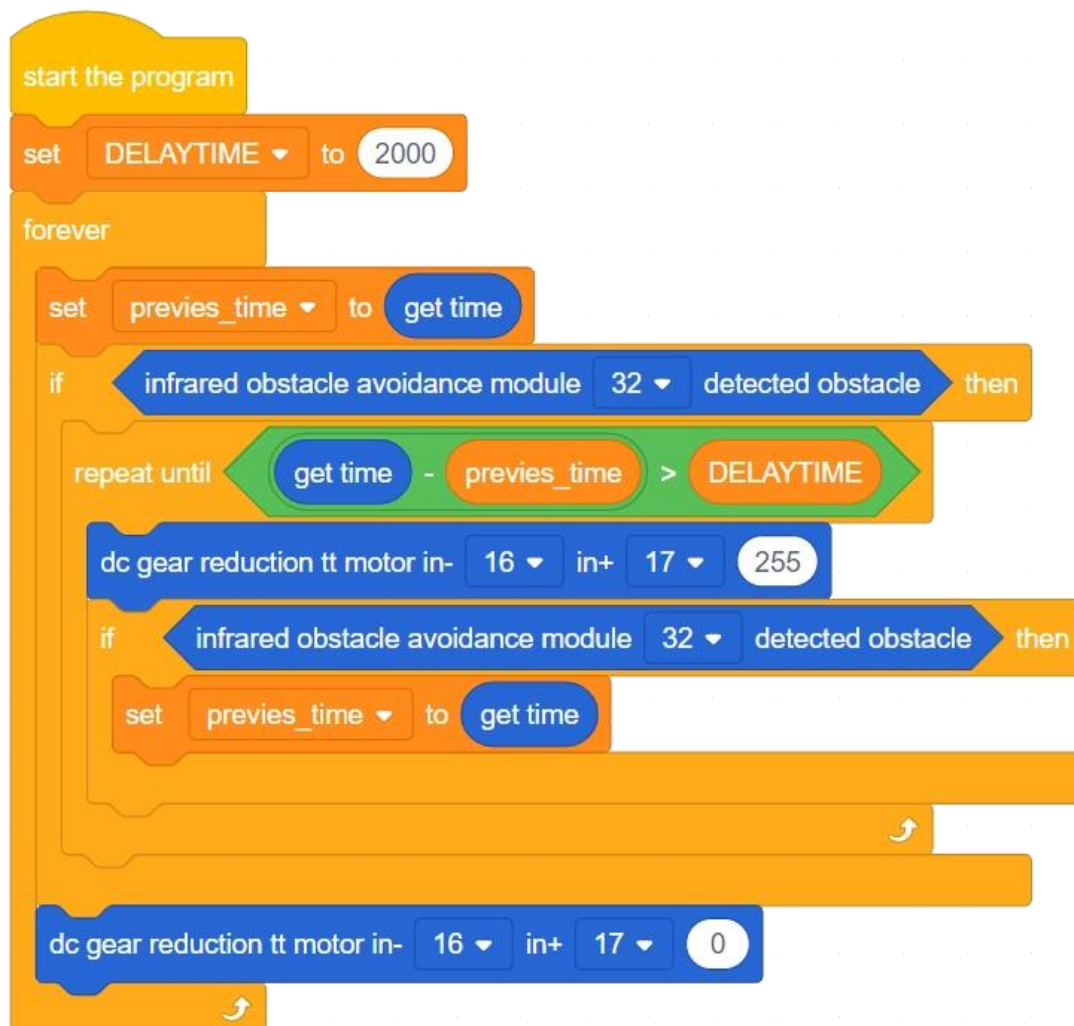


1. Program Flowchart





2. Coding





3. Instruction Parsing

infrared obstacle avoidance module 32 ▼ detected obstacle

: This command

is in the "Robots" instruction module. You can set the pin connected to the infrared obstacle avoidance sensor in the dropdown box. This command can detect whether the sensor has detected an obstacle. It returns True when an obstacle is detected, and False otherwise.

get time

: This command is in the "Robots" instruction module. It is used to obtain the time that the development board has been running the current program, representing the number of milliseconds that have passed from when the development board started running the current program until the function is called.



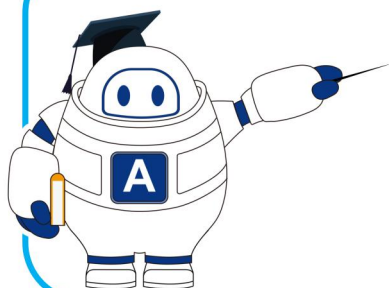


Section 4 Smart Sorting



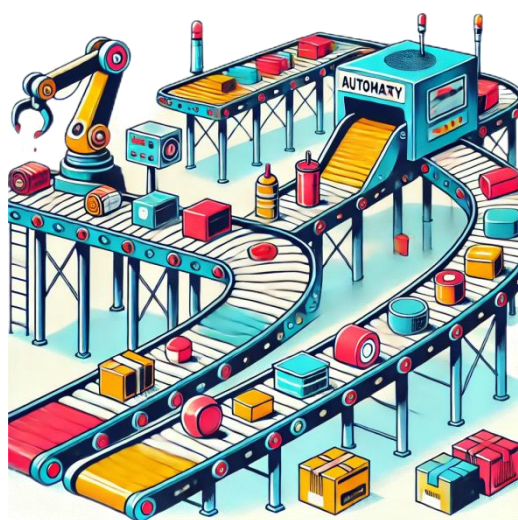
Course Objects

- Understand the role of smart sorting systems in assembly lines.
- Learn the principles of color sensor.
- Master the use of color sensor.
- Learn how to build a smart sorting system.



How does a smart assembly line sort out products that are not of acceptable quality?

In smart assembly lines, automatic product sorting plays a crucial role, significantly enhancing production efficiency and product quality. Given the variety of products on the production line, it is essential to differentiate products by model and specifications for targeted packaging and transportation. Additionally, the sorting functionality of the assembly line





assists in product quality inspection and sorting, ensuring that defective products are effectively filtered out.

How are different types of products categorized? Smart sorting systems continuously monitor products passing on the assembly line using sensors, cameras, or other detection devices. These systems can identify various product characteristics, such as size, shape, color, weight, barcodes, and QR codes. After these data are collected, the system categorizes products based on pre-set classification standards.

In this lesson, we will use a color sensor to simulate a smart sorting system.



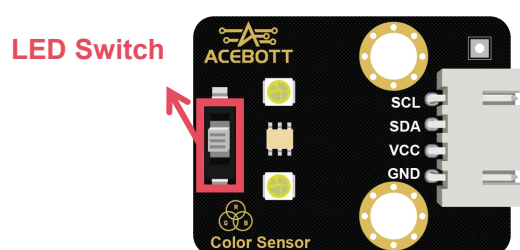
A color sensor is a device used to detect and identify the colors of objects by measuring the components of red, green, and blue light emitted from the object's surface. It is commonly used in industrial automation, robotics technology, smart homes, and consumer electronics.

In this lesson, we are using the TCS3472 model of the color sensor, which can measure the light values of red, green, and blue colors. These data can be used to determine the color of an object. Additionally, the sensor can measure the intensity of clear light, which is useful for ambient light sensing. The sensor





is equipped with an I2C interface, which allows the measured data to be transmitted to the controller via I2C communication. The TCS3472 features a built-in infrared blocking filter to reduce the interference of infrared light on measurements, thereby enhancing measurement accuracy. This sensor also has an integrated LED light source, which can be controlled by a switch to provide illumination for color measurement, further improving accuracy.



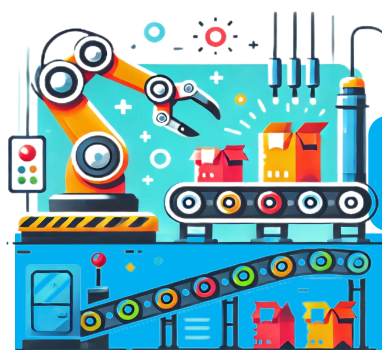
Color Sensor - Pin Analyse

SCL: Connect the SCL pin of the controller board.

SDA: Connect the SDA pin of the controller board.

VCC: Connect the VCC pin of the controller board.

GND: Connect the GND pin of the controller board.



Smart Factory Lab

Smart Sorting



Project Description



Use a conveyor belt to transport objects and sort them into different areas based on their color.



Hardware Scheme

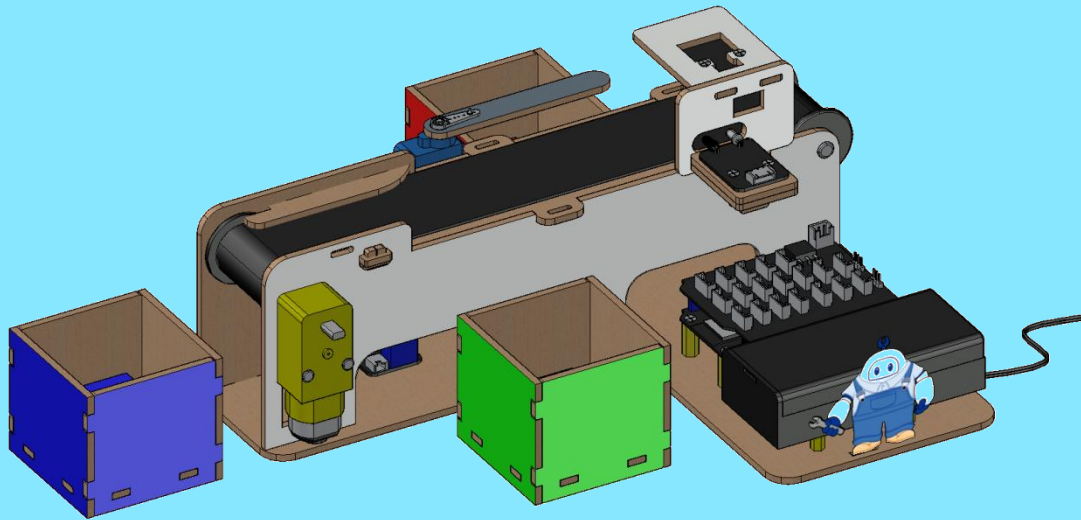
1. Hardware List

Picture	Name	Quantity
	ESP32 Board	1
	ESP32 Expansion Board	1
	TT DC Motor	1
	DC Motor Driver Module	1
	Infrared Obstacle Avoidance Sensor	1
	Color Sensor	1
	Servo SG90	1
	Type-C Data Cable	1



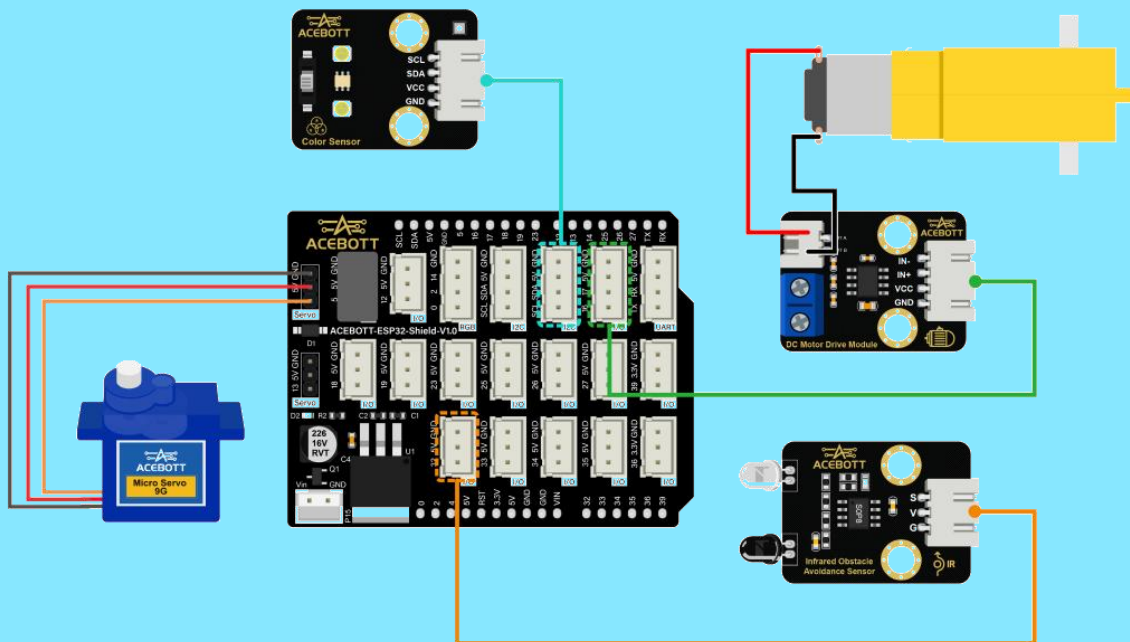


2. Hardware Structure



For detailed assembly steps, refer to Section 11 of the assembly document, titled "11 Smart Sorting".

3. Hardware Wiring

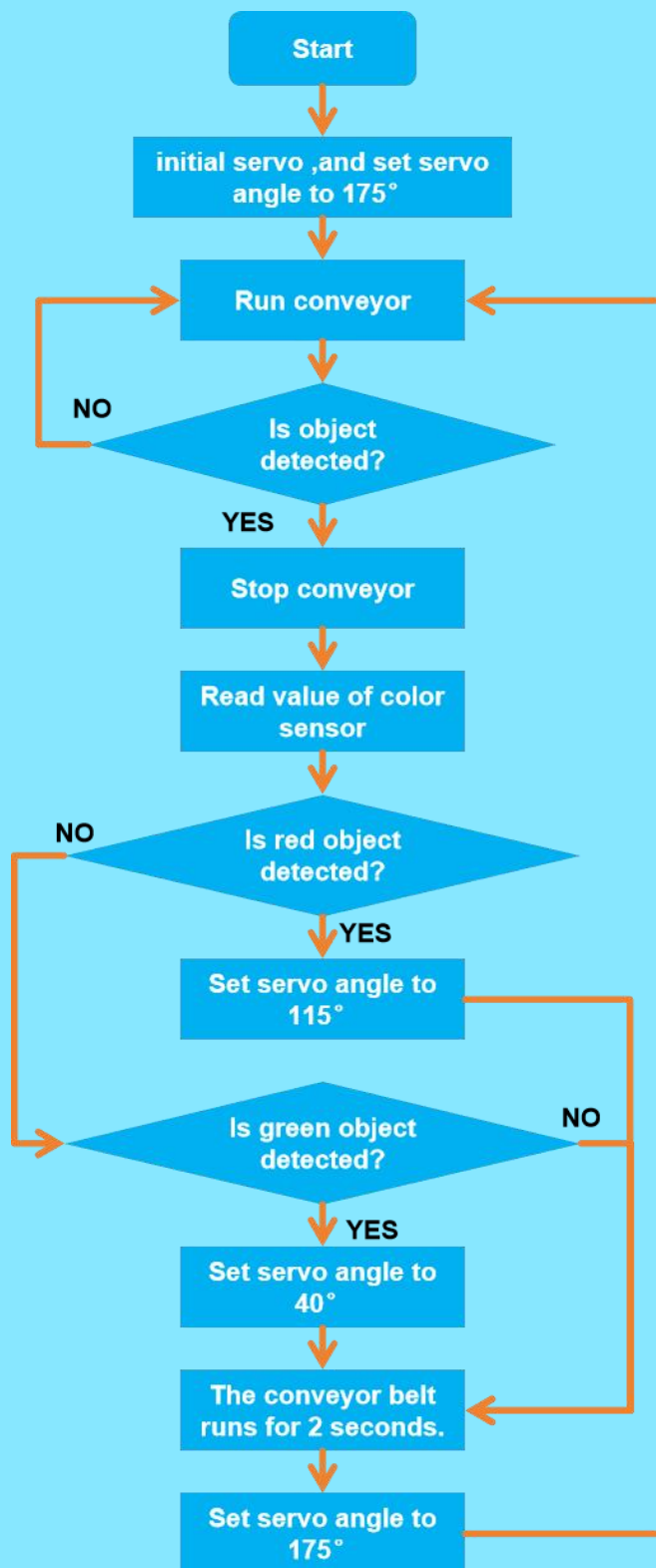




Programming

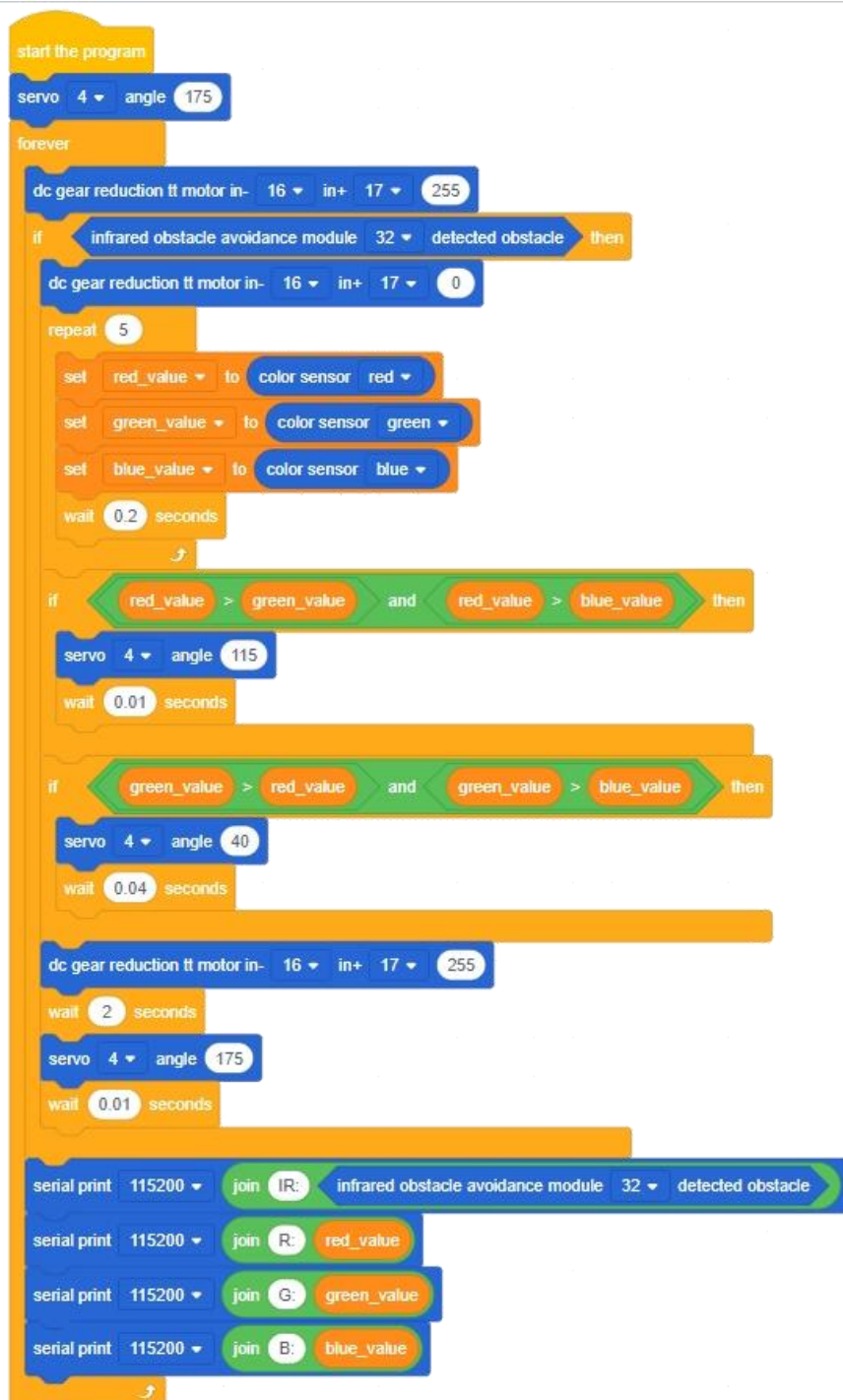


1. Program Flowchart





2. Coding





3. Instruction Parsing

color sensor brightness ▼

: This command is within the "Robots" instruction module. It can obtain the color and brightness values detected by the color sensor, and you can select different parameters by clicking on the dropdown box.





Chapter 4 Smart Warehouse

The warehouse system is an indispensable part of a factory. It not only connects the production line, ensuring the timely supply of raw materials and components, but also directly faces the market. It needs to quickly respond to changes in customer orders and fluctuations in market demand to adjust product inventory. Smart warehousing integrates advanced automation technologies and information technology into the warehouse system to achieve efficient, precise, and automated warehouse operations and management. As a key component of modern logistics and supply chain management, smart warehousing can significantly enhance storage efficiency, reduce operational costs, and optimize inventory management.



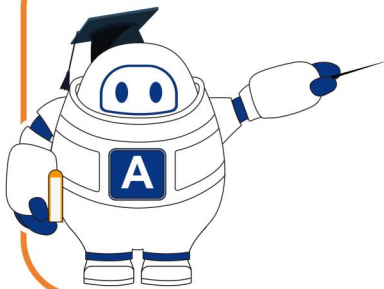


Section 1 Smart Storage



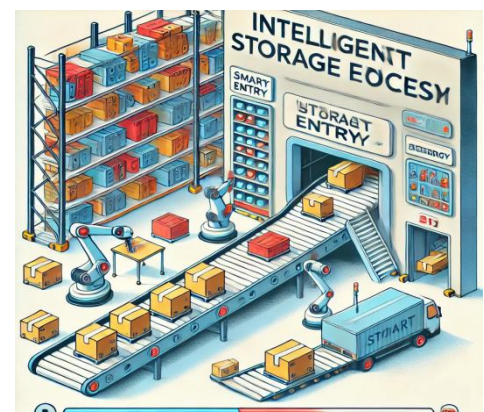
Course Objects

- Understand the process of smart storage
- Learn the principles of RFID technology.
- Master the use of RFID module.
- Build a smart warehousing system and implement smart storage.



In smart warehousing, how are products accurately stored in the correct storage locations?

The warehousing operation in smart warehousing refers to the process of receiving materials, semi-finished, or finished products from external sources and storing them safely and efficiently in the warehouse. In smart warehousing systems, the warehousing



operations are typically automated and intelligent, integrating technologies such as the Internet of Things, automation equipment, and barcode/RFID





recognition to achieve efficient inventory management. Here is the basic process for smart warehousing:

1. Goods Receipt: Before goods arrive at the warehouse, the system already receives incoming stock information from suppliers or production lines, including details like the type, quantity, and batch of materials. Automated equipment is then used to transport the goods to the warehouse's receiving area.

2. Automatic Identification: As goods pass through the receiving area, automatic identification devices scan the RFID tags or barcodes on the goods to capture and verify the information.

3. Allocation of Storage Location: The warehouse management system automatically assigns an appropriate storage location based on factors such as the type of goods, quantity, storage requirements, and inventory strategies.

4. Automated Storage: Based on the assigned location, goods are transported from the receiving area to the specified storage location using automated equipment.

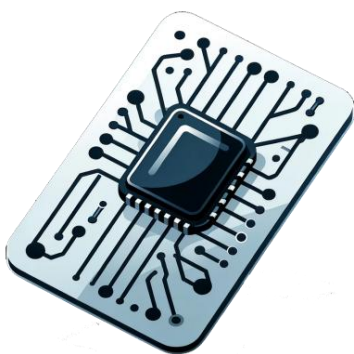
In this lesson, we will simulate smart storage in smart warehousing using a combination of conveyor belt, robotic arm, and RFID module.





RFID, short for Radio-Frequency Identification, is an automatic identification technology that utilizes wireless radio signals to identify targets and retrieve relevant data. Its basic principle involves using electromagnetic fields and radio frequency communication to wirelessly transmit information stored in RFID tags to RFID readers, enabling tracking, identification, and management of target objects. RFID technology finds wide applications in fields such as logistics management, inventory tracking, asset management, intelligent transportation, and identity recognition.

The RFID system typically consists of two main components: RFID tags, RFID readers/writers.



RFID tags are electronic tags equipped with chips and antennas, which can achieve object recognition and data storage. When the RFID tag approaches the RFID reader, the reader generates a certain frequency and power RF signal to activate the RFID

tag. After activation, the RFID tag will receive RF signal energy and use its working circuit for demodulation and data processing, and then return the data stored in the chip to the reader/writer through the RF signal.

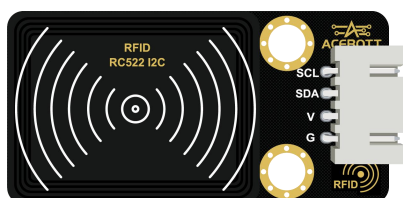




RFID readers/writers is a device used to read and write data from RFID tags. It can send radio frequency signals and receive data returned by RFID tags, and interact with RFID tags through radio frequency communication. Readers typically include components such as antennas, RF modules, processors, etc., which can recognize RFID tags, read and write data.



In this lesson, we are using the RFID RC522 IIC module, a common radio frequency identification module that can wirelessly identify and read/write data to nearby electronic tags. It communicates with the main control board or microcontroller via the IIC interface.



RFID Module - Pin Analyse

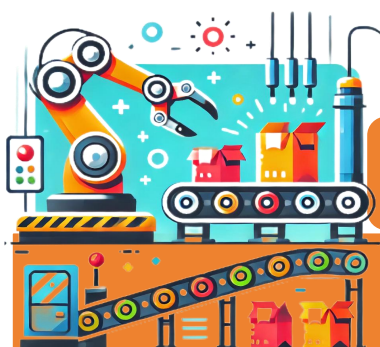
SCL: Connect the SCL pin of the controller board.

SDA: Connect the SDA pin of the controller board.

VCC: Connect the VCC pin of the controller board.

GND: Connect the GND pin of the controller board.





Smart Factory Lab

Smart Storage



Project Description

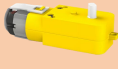


The process of realizing intelligent warehousing involves using a conveyor belt for the transportation of items. The items are classified using an RFID module. When the items reach the receiving area, the robotic arm automatically grabs the items and places them on the corresponding location on the shelf. Use a conveyor belt to transport objects and sort them into different areas based on their color.



Hardware Scheme

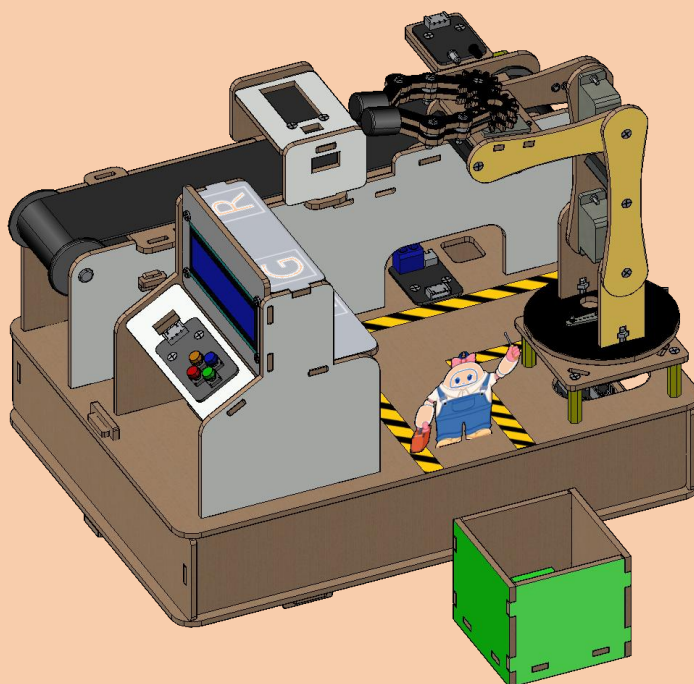
1. Hardware List

Picture	Name	Quantity
	ESP32 Board	1
	ESP32 Expansion Board	1
	TT DC Motor	1
	DC Motor Driver Module	1
	Infrared Obstacle Avoidance Sensor	1
	RFID Module	1
	MG90S Servo	4
	Type-C Data Cable	1



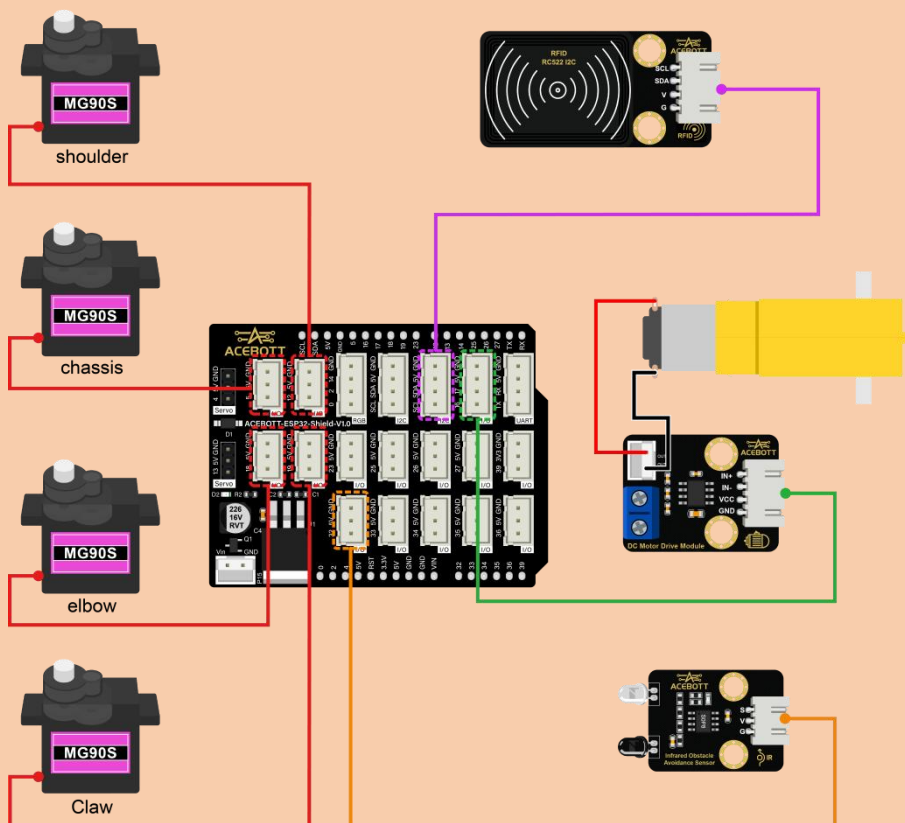


2. Hardware Structure



For detailed assembly steps, refer to Section 12 of the assembly document, titled "12 Smart Storage".

3. Hardware Wiring

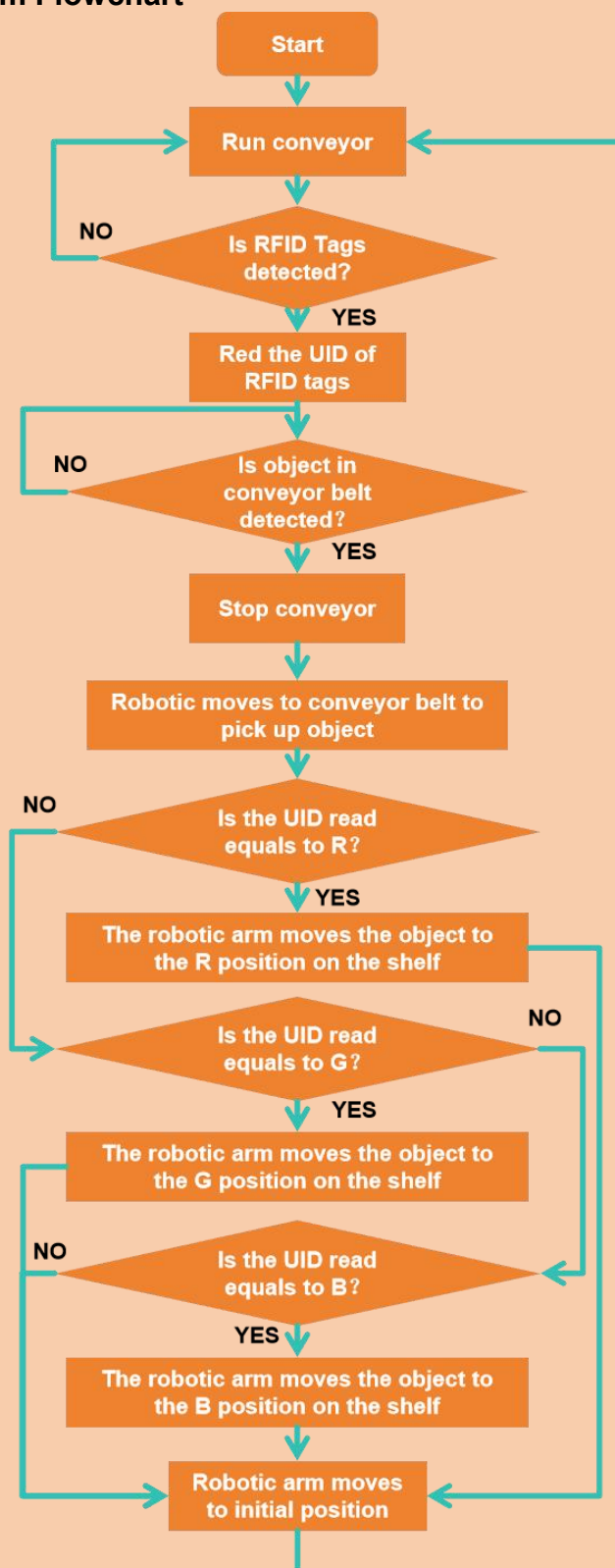




Programming

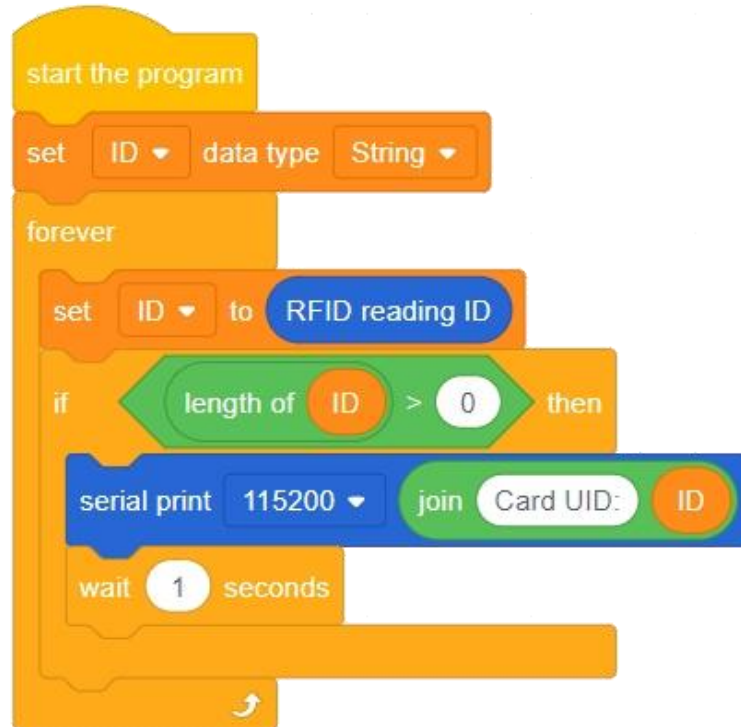


1. Program Flowchart





2.Coding1-Get the tag ID on the block



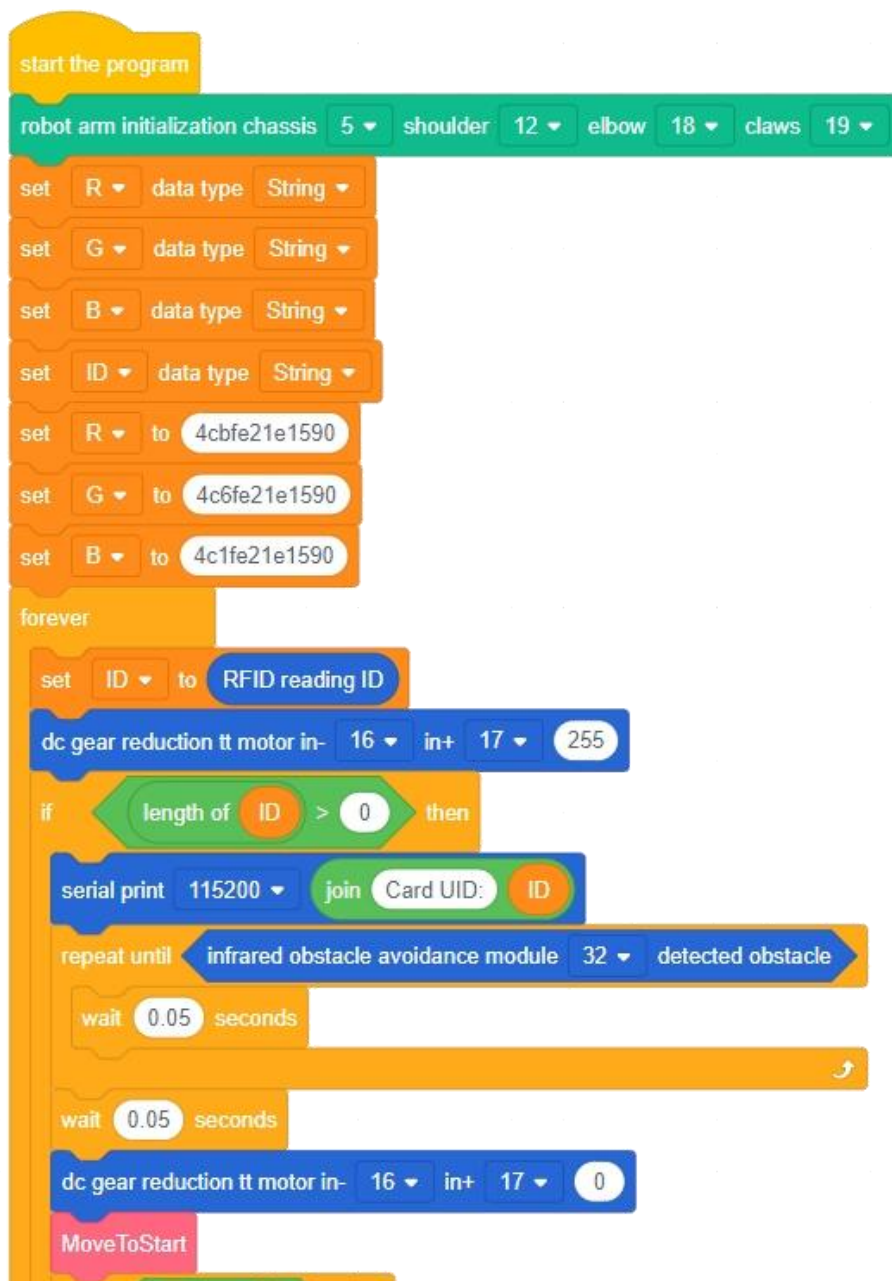
Tips

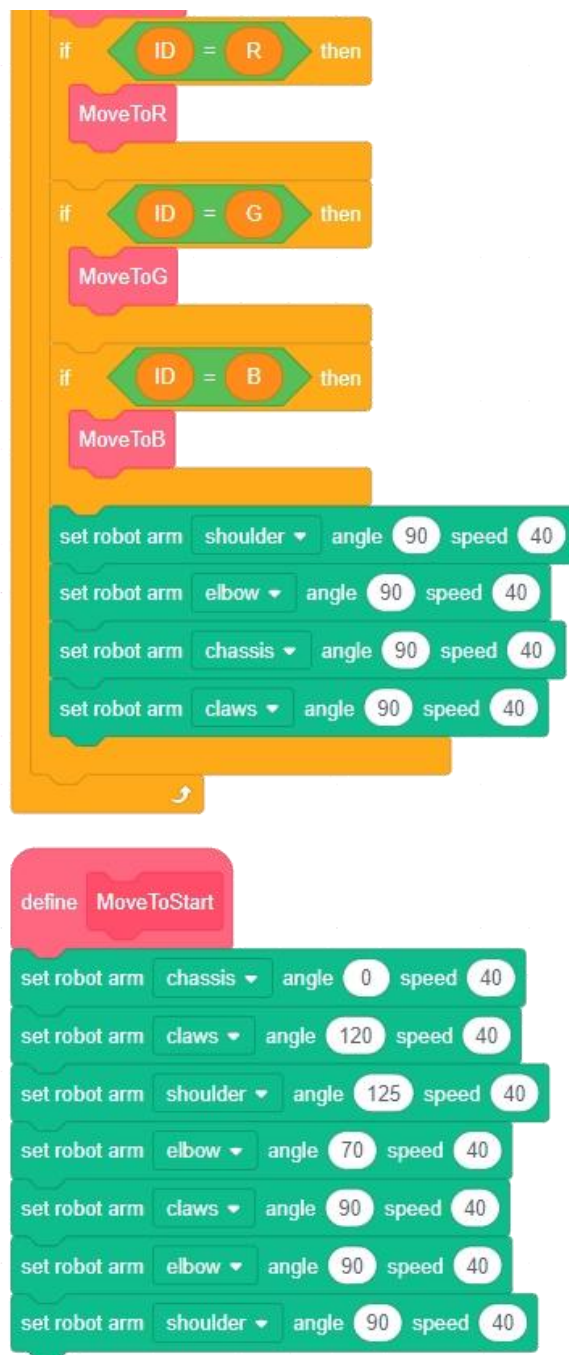
The tag ID of each color block is not exactly the same, and there may be different. You can confirm the tag ID of each color block through this step, and record the ID displayed by the serial port monitor for the next project program. For example, I get the red tag ID: 4cbfe 21e1590; green tag ID: :4c 6fe 21e1590; blue tag ID: :4c1fe 21e1590 through this step, you need to adjust according to your test results.





2. Coding2-Project Coding







define MoveToR

set robot arm chassis ▾ angle 85 speed 40

set robot arm elbow ▾ angle 125 speed 40

set robot arm shoulder ▾ angle 140 speed 40

wait 0.5 seconds

set robot arm claws ▾ angle 120 speed 40

define MoveToG

set robot arm chassis ▾ angle 105 speed 40

set robot arm elbow ▾ angle 115 speed 40

set robot arm shoulder ▾ angle 135 speed 40

wait 0.5 seconds

set robot arm claws ▾ angle 120 speed 40

define MoveToB

set robot arm chassis ▾ angle 95 speed 40

set robot arm elbow ▾ angle 80 speed 40

set robot arm shoulder ▾ angle 135 speed 40

wait 0.5 seconds

set robot arm claws ▾ angle 120 speed 40





3. Instruction Parsing

RFID reading ID

: This command is within the "Robots" instruction module. It can obtain the RFID tag information read by the RFID sensor. If an RFID tag is read, it will return the UID of that tag. If no RFID tag is read, it will return a null value. We can determine whether the RFID module has read an RFID tag by checking the character length of this data. If its character length is greater than 0, an RFID tag has been read; otherwise, no tag has been read.

set robot arm chassis ▼ angle 90 speed 100

: This command is in the "Robot Arm" extension. It can control the joints of the robotic arm to move to a specified angle at a designated speed, with a speed range of 0 to 100. The larger the value, the faster the robotic arm moves.

length of apple

: This command is in the "Operators" instruction module. It can be used to obtain the number of characters in a string.



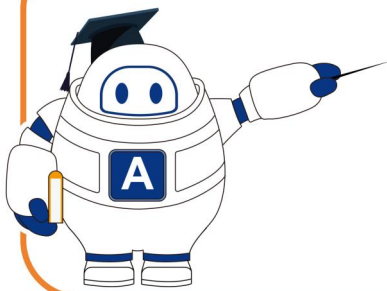


Section 2 Smart Outbound Logistics



Course Objects

- Understand the process of outbound logistics in smart warehousing.
- Learn the principles of a 4-digit keypad.
- Master the use of a 4-digit keypad.



How are products accurately retrieved from warehouse in smart factory?

Smart warehousing outbound operations involve retrieving materials, semi-finished products, or finished products from storage locations and transporting them to production lines, shipping areas, or other specified destinations. Like inbound operations, smart warehousing outbound operations utilize



automation equipment and intelligent systems to achieve efficient and precise





materials management. Here are the main steps in the outbound operations of smart warehousing:

1. Receiving Outbound Instructions: When shipments, production, or inventory transfers are needed, the system receives an outbound request. These requests may come from production plans, customer orders, or inventory scheduling needs within the warehouse management system.

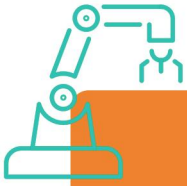
2. Determining the Outbound Location: Based on the requirements of the order, identify the type and model of the goods to be dispatched, and then determine their storage location within the warehouse.

3. Automated Retrieval: Once the location of the goods is known, automated equipment (such as palletizers, AGVs, robotic arms, etc.) travels to the specified storage location to retrieve the goods to be dispatched from the racks.

4. Transport to Shipping Area: After the goods are retrieved, automated equipment transports the goods to the shipping or loading area.

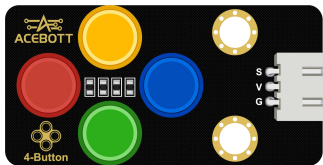
In this lesson, we will use a 4-digital keypad module to simulate the order functionality in the outbound process, where pressing different buttons will simulate different outbound instructions.





4-Button Module

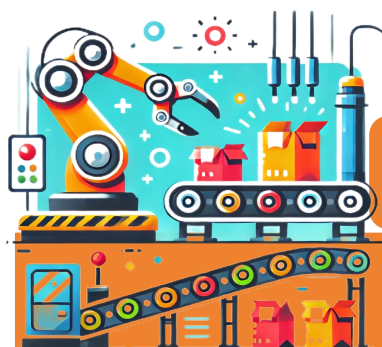
The 4-button module is an input device with four independent keys. When different keys are pressed by the user, the resistance values within the circuit change, leading to a change in the corresponding voltage output. Since this module outputs an analog signal, its signal pin must be connected to the analog input port of the controller. Note that the VCC of the module requires a 3.3V voltage input.



4-Button Module - Pin Analyse

- S:** Signal output pin, connects to the I/O pin of controller board.
- V:** Connects to the 3.3V pin of controller board.
- G:** Connects to the GND pin of controller board.





Smart Factory Lab

Smart Outbound Logistics



Project Description



implement the intelligent outbound process for smart warehousing. By pressing keys of different colors, the robotic arm picks up corresponding objects from the shelves and transports them to the shipping area.



Hardware Scheme

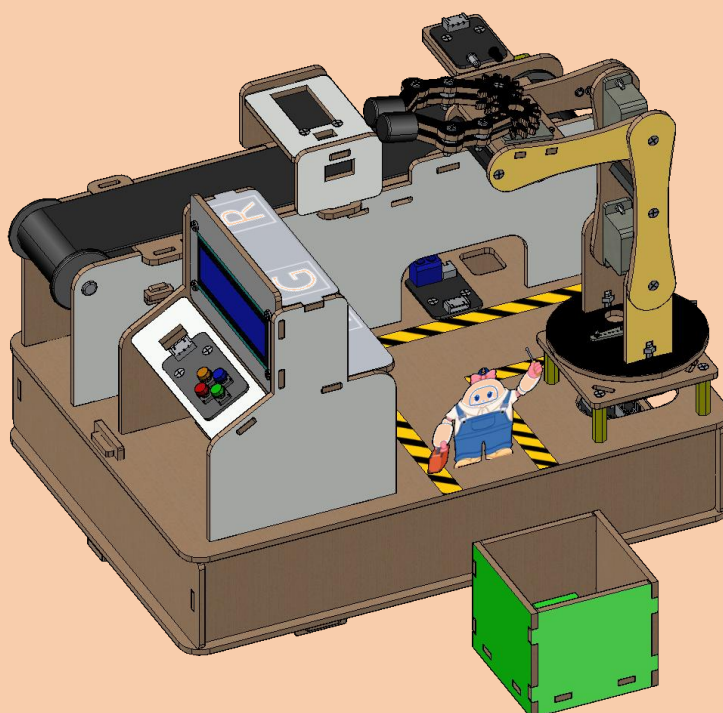
1. Hardware List

Picture	Name	Quantity
	ESP32 Board	1
	ESP32 Expansion Board	1
	4-Button Module	1
	MG90S Servo	4
	Type-C Data Cable	1



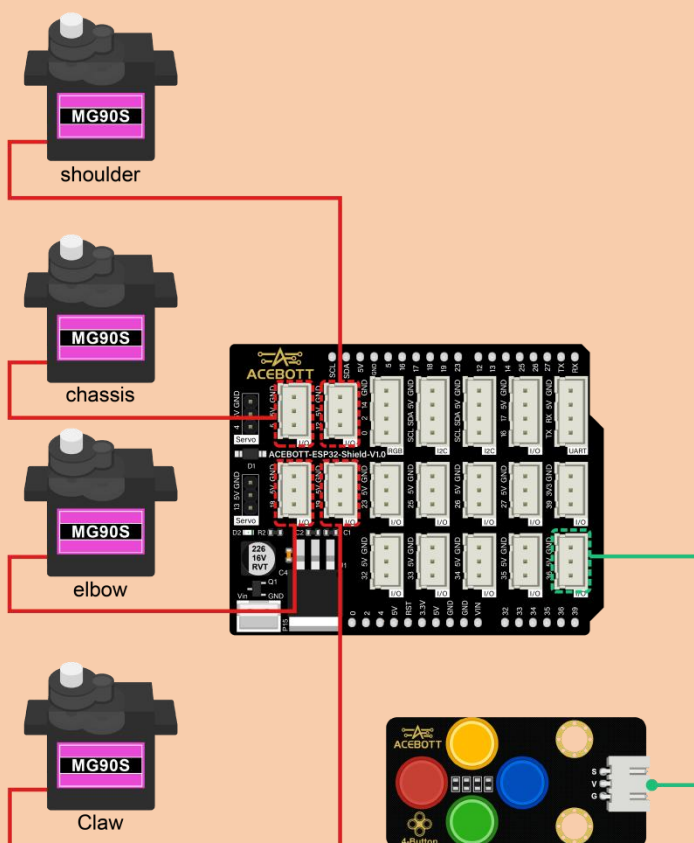


2. Hardware Structure



For detailed assembly steps, refer to Section 13 of the assembly document, titled "13 Smart Outbound Logistics".

3. Hardware Wiring

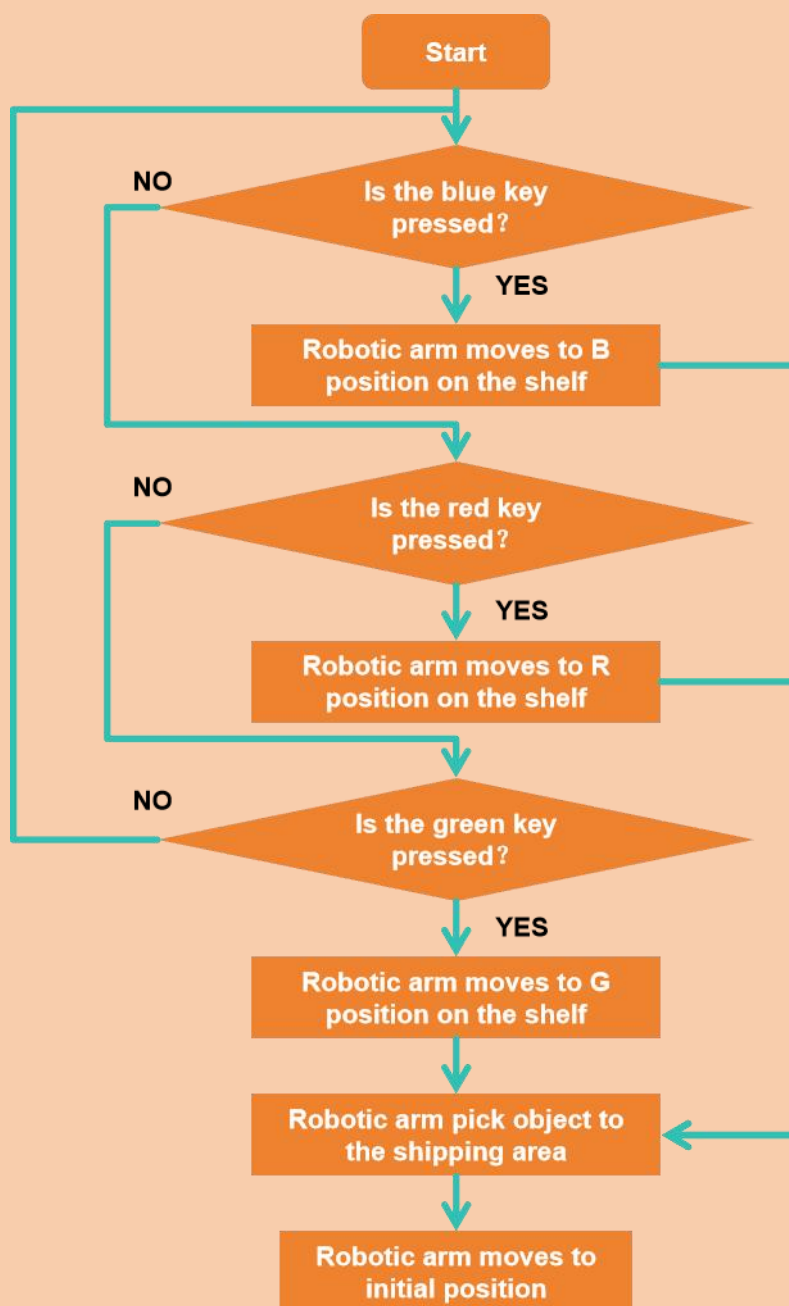




Programming

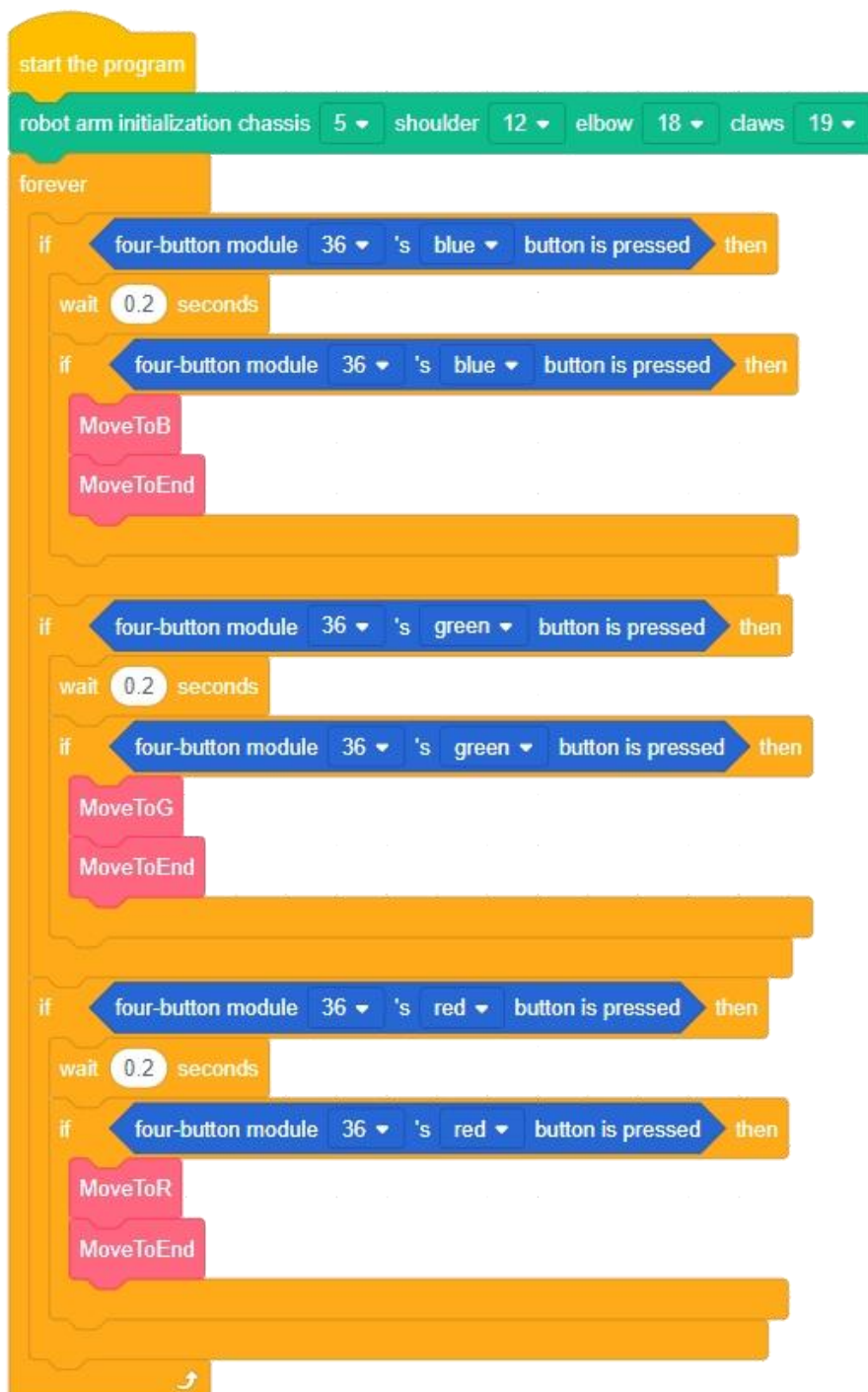


1. Program Flowchart





2. Coding





```
define MoveToR
  set robot arm claws ▾ angle 120 speed 40
  set robot arm chassis ▾ angle 75 speed 40
  set robot arm elbow ▾ angle 130 speed 40
  set robot arm shoulder ▾ angle 145 speed 40
  set robot arm claws ▾ angle 90 speed 40
  set robot arm shoulder ▾ angle 90 speed 40
  set robot arm elbow ▾ angle 90 speed 40
```

```
define MoveToG
  set robot arm claws ▾ angle 120 speed 40
  set robot arm chassis ▾ angle 105 speed 40
  set robot arm elbow ▾ angle 115 speed 40
  set robot arm shoulder ▾ angle 135 speed 40
  set robot arm claws ▾ angle 90 speed 40
  set robot arm shoulder ▾ angle 90 speed 40
  set robot arm elbow ▾ angle 90 speed 40
```



define MoveToB

set robot arm claws ▾ angle 120 speed 40

set robot arm chassis ▾ angle 85 speed 40

set robot arm elbow ▾ angle 80 speed 40

set robot arm shoulder ▾ angle 135 speed 40

set robot arm claws ▾ angle 90 speed 40

set robot arm shoulder ▾ angle 90 speed 40

set robot arm elbow ▾ angle 90 speed 40

define MoveToEnd

set robot arm chassis ▾ angle 175 speed 40

set robot arm shoulder ▾ angle 125 speed 40

set robot arm elbow ▾ angle 70 speed 40

set robot arm claws ▾ angle 120 speed 40

set robot arm elbow ▾ angle 90 speed 40

set robot arm shoulder ▾ angle 90 speed 40

set robot arm chassis ▾ angle 90 speed 40





3. Instruction Parsing

four-button module 36 ▼ 's blue ▼ button is pressed

: This command is within the "Robots" instruction module. It allows you to determine whether a button on the 4-Button Module has been pressed, and you can select different buttons by clicking on the dropdown box.



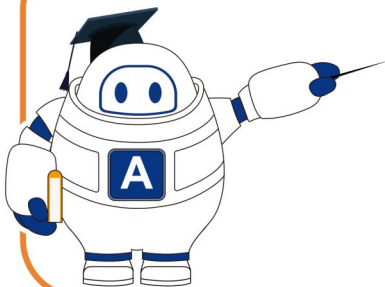


Section 3 Smart Warehousing Integrated System



Course Objects

- Understand the process of the Smart Warehousing Integrated System.
- Learn the principles of the LCD1602 display.
- Master the use of the LCD1602 display.
- Build a smart warehousing system that integrates smart storage, smart outbound logistics, and inventory management functions.



In smart warehousing, how can you know the real-time information about the goods in the warehouse?

In our previous studies, we have gained a preliminary understanding of the storage and outbound processes within the smart warehousing system. smart storage and outbound logistics are core functionalities of a smart warehousing system that effectively





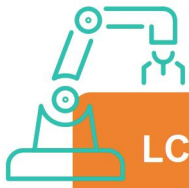
enhance storage efficiency and accuracy. However, beyond implementing these basic functionalities, a smart warehousing system also needs the capability to monitor goods information in real-time.

By monitoring goods information in real-time, the smart warehousing system can promptly access the latest status of inventory, ensuring the accuracy and timeliness of inventory data. This real-time monitoring not only helps warehouse managers better understand inventory conditions but also allows for the immediate identification and resolution of issues during the goods inbound and outbound processes, avoiding inventory discrepancies.

Additionally, the smart warehousing system can automatically adjust inventory levels based on changes in market demand. By analyzing market data and historical sales records, the system can forecast future market demands, thereby adjusting inventory in advance to ensure that stock levels match market needs. This not only reduces inventory backlog but also prevents sales losses due to stock shortages.

In this lesson, we will build a complete smart warehousing system and use an LCD1602 display to show inventory information in real-time.

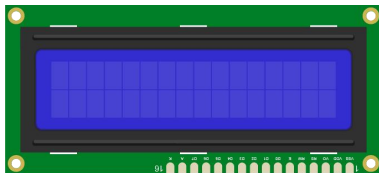




LCD1602 Display Module

The LCD1602 liquid crystal display is a dot matrix liquid crystal display module specifically designed to show letters, numbers, symbols, and more. It features two lines, each capable of displaying 16 characters. Each character is formed by a 5x7 dot matrix and can display standard ASCII characters and some other built-in characters, with space for eight custom characters. In this tutorial, the LCD1602 module is equipped with an I2C interface, allowing data communication with the controller via the I2C protocol, which significantly saves the controller's pin resources.

LCD1602 Display Module - Pin Analyse



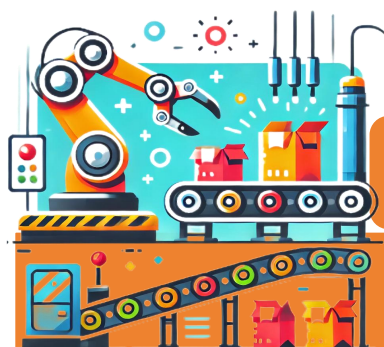
SCL: Connect the SCL pin of the controller board.

SDA: Connect the SDA pin of the controller board.

VCC: Connect the VCC pin of the controller board.

GND: Connect the GND pin of the controller board.





Smart Factory Lab

Smart Warehousing Integrated System



Project Description



Build smart warehousing system that integrates smart outbound logistics, smart storage logistics, and inventory monitoring functions.



Hardware Scheme

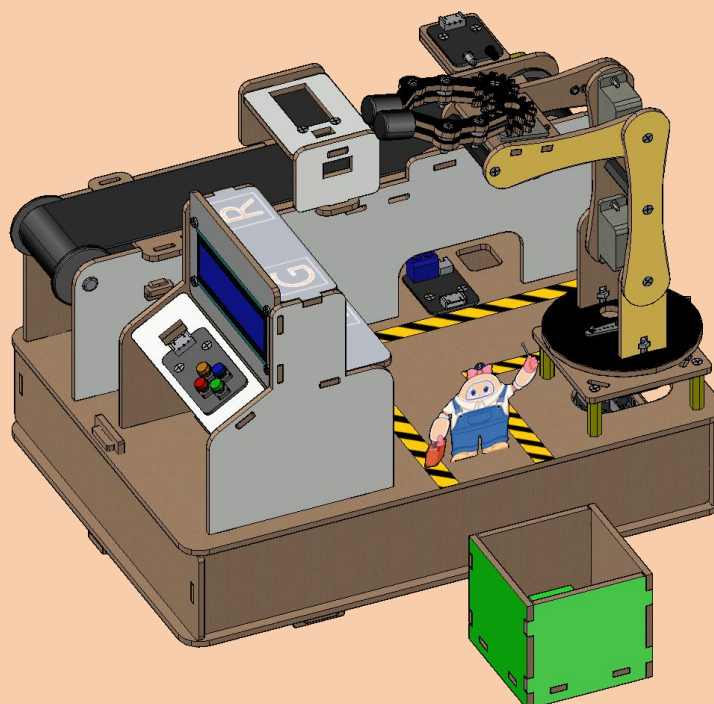
1. Hardware List

Picture	Name	Quantity
	ESP32 Board	1
	ESP32 Expansion Board	1
	TT DC Motor	1
	DC Motor Driver Module	1
	Infrared Obstacle Avoidance Sensor	1
	RFID Module	1
	MG90S Servo	4
	4-Button Module	1
	LCD1602 Module	1
	Type-C Data Cable	1



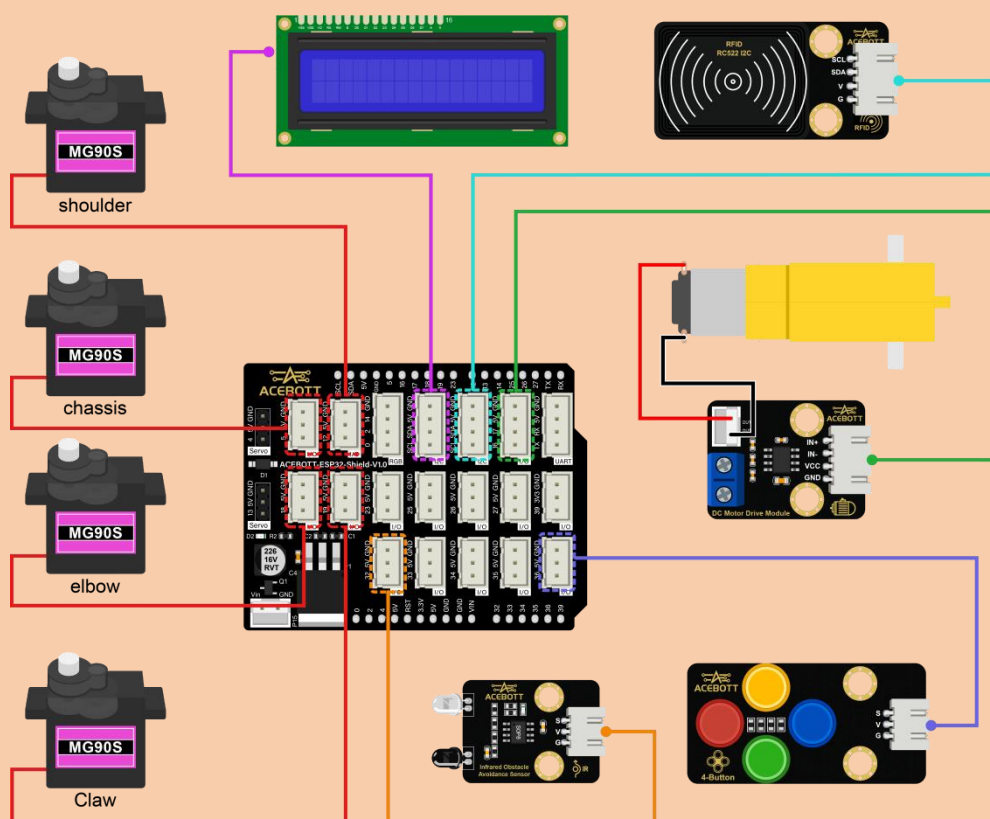


2. Hardware Structure



For detailed assembly steps, refer to Section 14 of the assembly document, titled "14 Smart Warehousing Integrated System".

3. Hardware Wiring

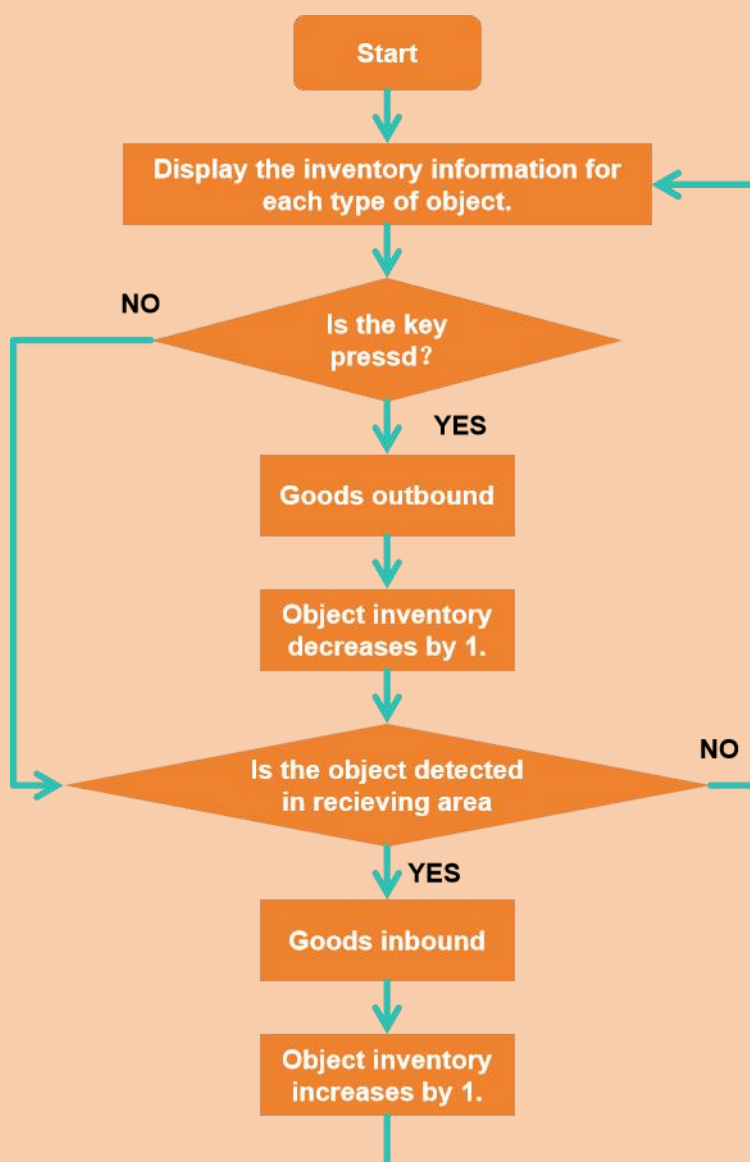




Programming

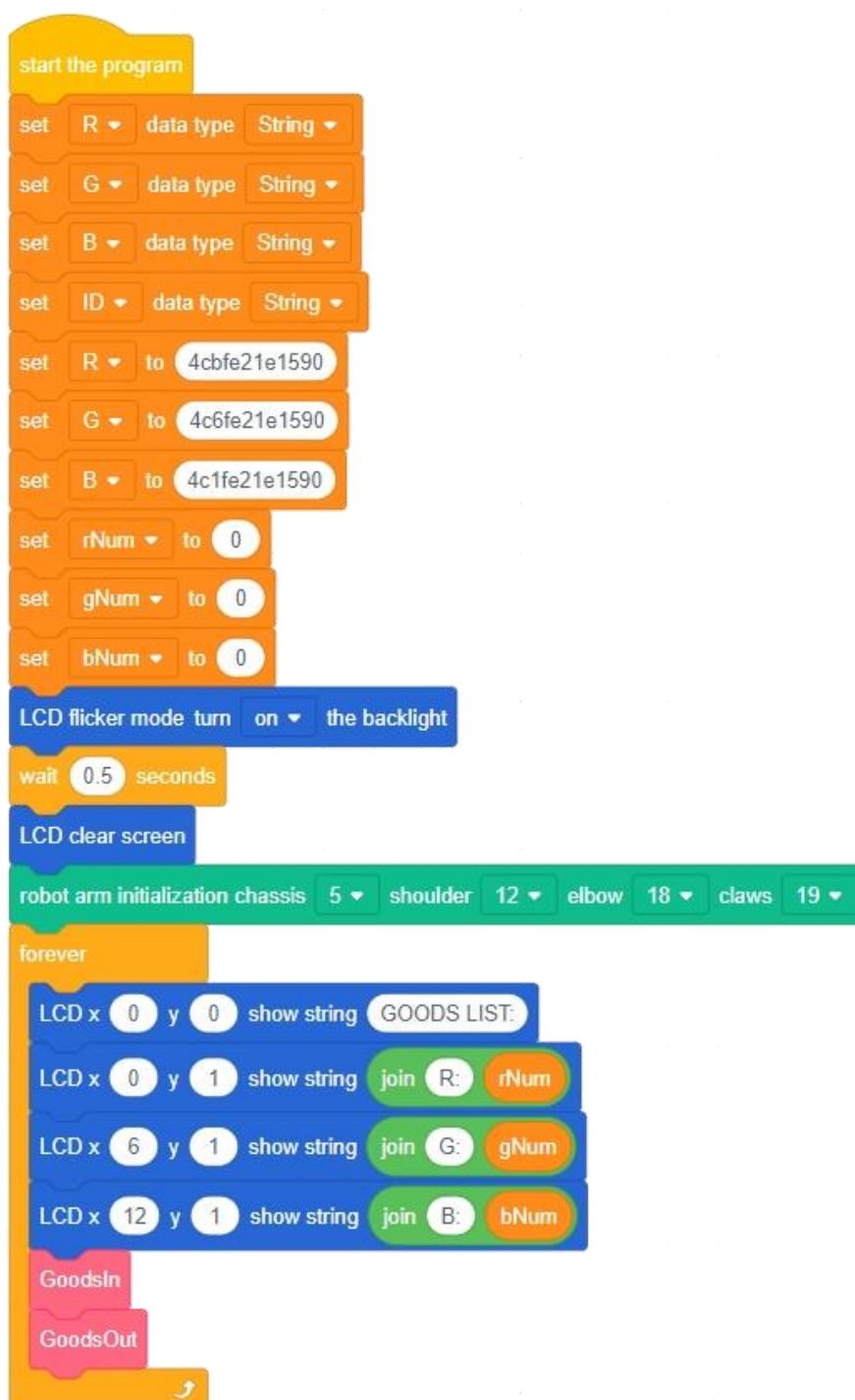


1. Program Flowchart





2. Coding





```
define GoodsIn
  dc gear reduction tt motor in- 16 in+ 17 255
  set ID to RFID reading ID
  if length of ID > 0 then
    serial print 115200 join Card UID: ID
    wait until infrared obstacle avoidance module 32 detected obstacle
    dc gear reduction tt motor in- 16 in+ 17 0
    MoveToStart
    if ID = R then
      MoveToR1
      change rNum by 1
    if ID = G then
      MoveToG1
      change gNum by 1
    if ID = B then
      MoveToB1
      change bNum by 1
    set robot arm shoulder angle 90 speed 40
    set robot arm elbow angle 90 speed 40
    set robot arm chassis angle 90 speed 40
    set robot arm claws angle 90 speed 40
```





define MoveToStart

```
set robot arm chassis ▾ angle 0 speed 40
set robot arm claws ▾ angle 120 speed 40
set robot arm shoulder ▾ angle 125 speed 40
set robot arm elbow ▾ angle 70 speed 40
set robot arm claws ▾ angle 90 speed 40
set robot arm elbow ▾ angle 90 speed 40
set robot arm shoulder ▾ angle 90 speed 40
```

define MoveToG1

```
set robot arm chassis ▾ angle 105 speed 40
set robot arm elbow ▾ angle 115 speed 40
set robot arm shoulder ▾ angle 135 speed 40
wait 0.5 seconds
set robot arm claws ▾ angle 120 speed 40
```

define MoveToR1

```
set robot arm chassis ▾ angle 85 speed 40
set robot arm elbow ▾ angle 130 speed 40
set robot arm shoulder ▾ angle 140 speed 40
wait 0.5 seconds
set robot arm claws ▾ angle 120 speed 40
```

define MoveToB1

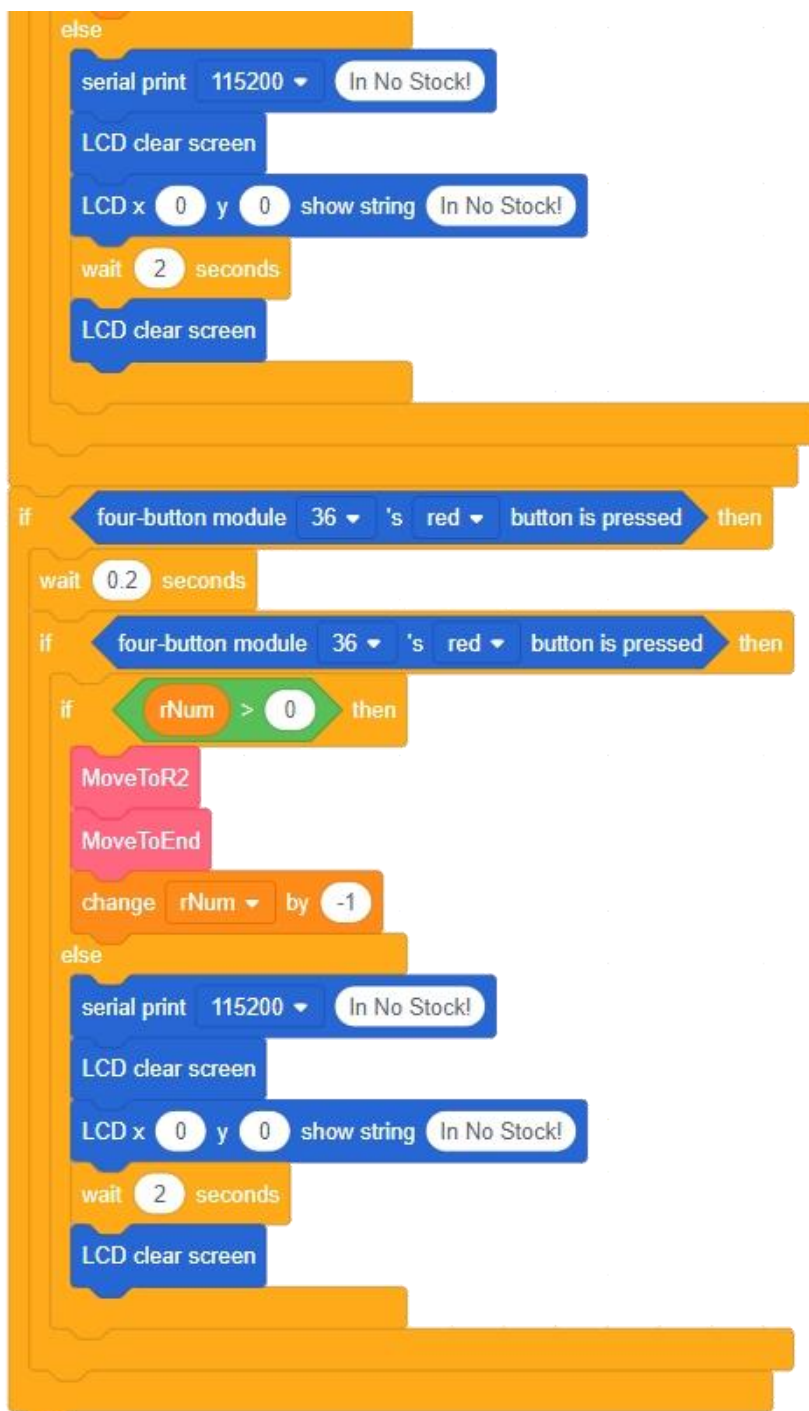
```
set robot arm chassis ▾ angle 95 speed 40
set robot arm elbow ▾ angle 80 speed 40
set robot arm shoulder ▾ angle 135 speed 40
wait 0.5 seconds
set robot arm claws ▾ angle 120 speed 40
```





```
define GoodsOut
  if four-button module 36 's blue button is pressed then
    wait 0.2 seconds
    if four-button module 36 's blue button is pressed then
      if bNum > 0 then
        MoveToB2
        MoveToEnd
        change bNum by -1
      else
        serial print 115200 In No Stock!
        LCD clear screen
        LCD x 0 y 0 show string In No Stock!
        wait 2 seconds
        LCD clear screen
    end if
  end if
  if four-button module 36 's green button is pressed then
    wait 0.2 seconds
    if four-button module 36 's green button is pressed then
      if gNum > 0 then
        MoveToG2
        MoveToEnd
        change gNum by -1
      else
    end if
  end if
```







define MoveToR2

```
set robot arm claws ▾ angle 120 speed 40
set robot arm chassis ▾ angle 75 speed 40
set robot arm elbow ▾ angle 130 speed 40
set robot arm shoulder ▾ angle 145 speed 40
set robot arm claws ▾ angle 90 speed 40
set robot arm shoulder ▾ angle 90 speed 40
set robot arm elbow ▾ angle 90 speed 40
```

define MoveToB2

```
set robot arm claws ▾ angle 120 speed 40
set robot arm chassis ▾ angle 85 speed 40
set robot arm elbow ▾ angle 80 speed 40
set robot arm shoulder ▾ angle 135 speed 40
set robot arm claws ▾ angle 90 speed 40
set robot arm shoulder ▾ angle 90 speed 40
set robot arm elbow ▾ angle 90 speed 40
```

define MoveToG2

```
set robot arm claws ▾ angle 120 speed 40
set robot arm chassis ▾ angle 105 speed 40
set robot arm elbow ▾ angle 115 speed 40
set robot arm shoulder ▾ angle 135 speed 40
set robot arm claws ▾ angle 90 speed 40
set robot arm elbow ▾ angle 90 speed 40
set robot arm shoulder ▾ angle 90 speed 40
```

define MoveToEnd

```
set robot arm chassis ▾ angle 175 speed 40
set robot arm shoulder ▾ angle 125 speed 40
set robot arm elbow ▾ angle 70 speed 40
set robot arm claws ▾ angle 120 speed 40
set robot arm elbow ▾ angle 90 speed 40
set robot arm shoulder ▾ angle 90 speed 40
set robot arm chassis ▾ angle 90 speed 40
```





3. Instruction Parsing

LCD clear screen

: This command is within the "Robots - LCD Module" instruction module. It can be used to clear the LCD module's buffer, effectively clearing the screen.

LCD flicker mode turn on ▼ the backlight

: This command is in the "Robots - LCD Module" instruction module. It is used to turn on or off the backlight of the LCD.

LCD x 0 y 0 show string Hi

: This command is within the "Robots - LCD Module" instruction module. It can be used to display specified character content at a specified location on the LCD screen.





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 QR Code



YouTube QR Code



ACEBOTT STEM Education Tech Co.,Ltd

0755-23956813

www.cebott.com

Room 306, Building 212, Tairan Science Park, Terra 4th Rd, Futian District, ShenZhen, China, 518042