Intelligent Logistics Car Based on PID Control

Dikai Ye, Qiuran Zhao, Tianyou Wang, Jialang Yang, Yifan Feng, Yuxiang Peng

[1] *South China University of Technology*
[2] *SHIEN-MING WU School of Intelligent Engineering*

June 8, 2022

Abstract

In order to meet the needs of engineering education and curriculum teaching reform, this paper designs and studies the obstacle avoidance and automatic grasping functions of the car based on Arduino MEGA2560. The automatic obstacle avoidance function is realized by the radar system composed of ultrasonic sensor and reducer motor, and a 6-DOF manipulator with Open MV or infrared sensor as module is designed. This paper introduces the overall operation process of the system in detail, describes the ultrasonic obstacle avoidance, the tracking PID control, the adjustment of the fixed position of the manipulator, the cases of different recognition objects and the relevant program algorithms, summarizes the shortcomings and puts forward more effective improvement methods. In the process of obstacle avoidance, we also designed two driving routes, and determined the better scheme according to the time cost, program complexity and turning efficiency. In the process of placing objects, we designed an intelligent push rod system with simple and efficient structure. The intelligent car has passed the experimental test, and its function meets the basic needs of obstacle avoidance, tracking, grasping and placing objects, which has a certain practical significance.

Keywords: Arduino, tracking, obstacle avoidance, ultrasonic sensor, 6-DOF Manipulator

Contents

1 Introduction

At present, the general multi axis manipulator is combined with the machine vision module to realize the function of target feature recognition and automatic handling. It is widely used in the handling and sorting occasions of industrial automation production line, and effectively solves the problems of excessive workload and increasing labor cost in manual sorting and handling. Moreover, the identification and extraction of target information is an important technical means of intelligent identification and automatic monitoring system[1], which has important research significance. At the same time, the automatic tracking obstacle avoidance vehicle can effectively transport industrial products to the designated place, which helps to realize the complete automation of the production line. Intelligent obstacle avoidance vehicle is the prototype of automatic driving vehicle and one of the necessary foundations. It mainly solves the problem of automatic obstacle avoidance during driving. The solution of this problem is also one of the necessary conditions for automatic driving of auxiliary vehicle. Therefore, it is of certain significance to study and explore it. However, the current general multi axis manipulator has a large size and is controlled by motion control card, which is expensive. Especially for general small sorting and handling application scenarios, the existing general manipulator is too conservative and bulky[2], which is not conducive to small enterprises to reduce economic costs. The combination of manipulator and intelligent obstacle avoidance car is only in the link of experiment. Therefore, this paper takes Arduino development board as the core controller, combined with the hardware design and program call of ultrasonic sensor, motor and steering gear, and designs a light manipulator control system with image recognition function to realize an automatic traveling car that can automatically recognize and grasp objects and has obstacle avoidance function.

2 Concept Design

The task is to design a smart vehicle which should be able to loading and unloading designated objects, carrying loads, tracking paths, and avoiding obstacles according to the map in Figure 1.

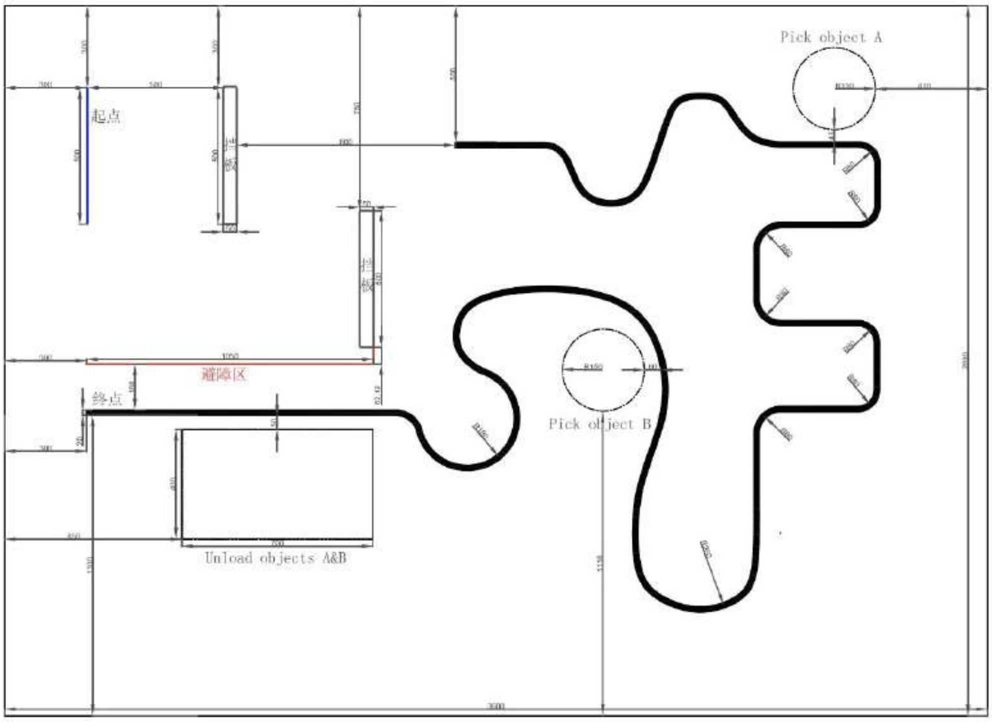


Figure 1: *Playground map*

The basic task requires us to pick up orange and eraser, addition points will be given if we pick up a bottle of water 600 ml and a pen. Clearly, different objects have different shape, size and weight. To complete all tasks, we are supposed to design a compatible loading device. For example, we might design a kind of manipulator that it can not only grasp large weight, but also close the claw angle very small. What's more, the speed of the intelligent car should not be too slow, because time is also a very important assessment requirement.

2.1 Intelligent Car System

This design selects Arduino microprocessor as the main control chip. It is a widely used electronic chip. Its development language is C language, which has the advantage of simple use [3]. The main feature of Arduino is the functional setting of its parameters. It reduces the threshold of development to a great extent, so users do not need to understand its structure from the bottom[4]. In addition, it can also be simply connected with various sensors and electronic components, which is convenient for development and function expansion. In this design, the smart car is equipped with ultrasonic sensor and steering gear as the core components of the obstacle avoidance module. It collects the corresponding information of the road conditions of the car under the control of Arduino, and feeds back the detected information to Arduino for processing. Arduino can control the car to make corresponding actions through calling programs, so as to realize the obstacle avoidance function. Figure 2 represent the overall circuit diagram.

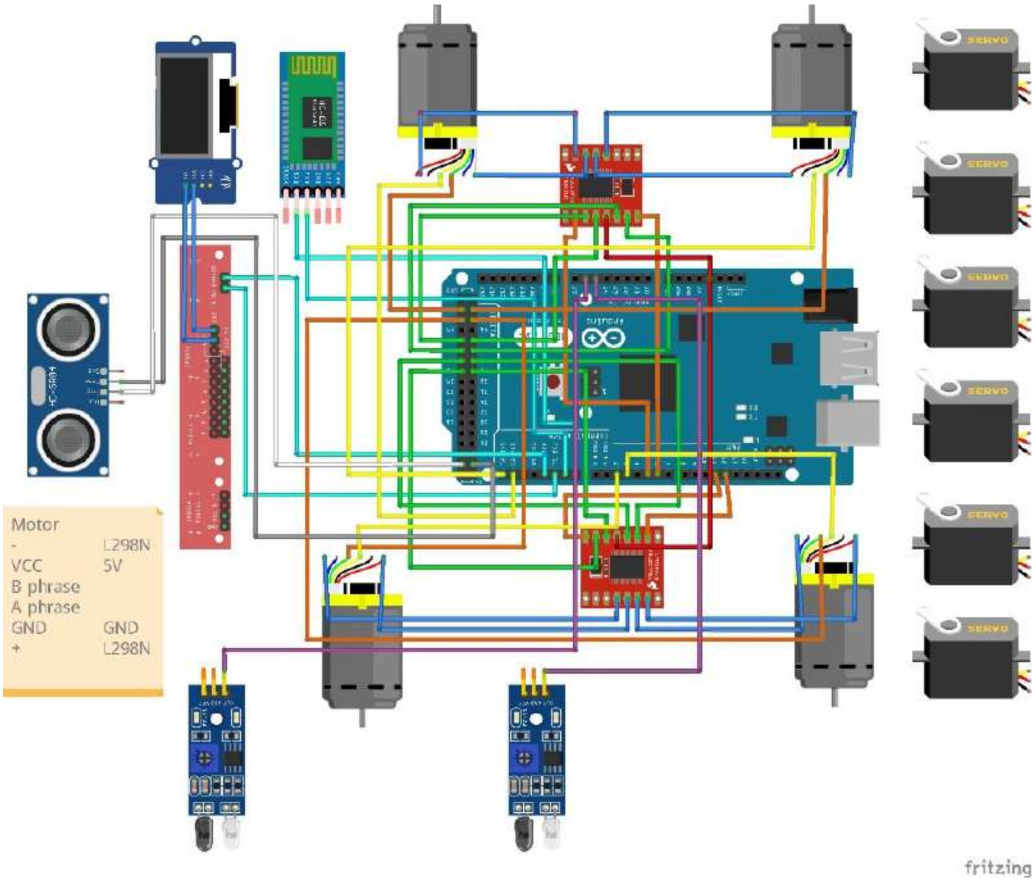


Figure 2: *Overall circuit diagram*

2.1.1 Power module

The power supply is an important part of the entire hardware circuit, it can provide a stable working voltage for the system circuit to make it work normally. Lithium battery has the characteristics of stable voltage and low price. In addition, it can also be charged and recycled. This design uses 12V lithium battery to power the car.

2.1.2 Motor, drive module and wheel

The four motors selected in this design are CHR-GM37-520 DC hall encoder reducer motor (Figure 3). Its advantages are all-metal gears, carbon brush permanent magnet motors, and support for forward and reverse rotation. The input voltage range of the motor is 6-24V. It adopts AB bi-phase Hall encoder and basic pulse 11PPR X gear reduction ratio, which meets the design conditions of intelligent car. At the same time, the Mecanum wheel is used, and the advantage is that it can be translated (Figure 4, 5). The motor drive module chooses TB6612FNG. Compared with the traditional L298N, the efficiency is greatly improved, and the volume is also greatly reduced (Figure 6).

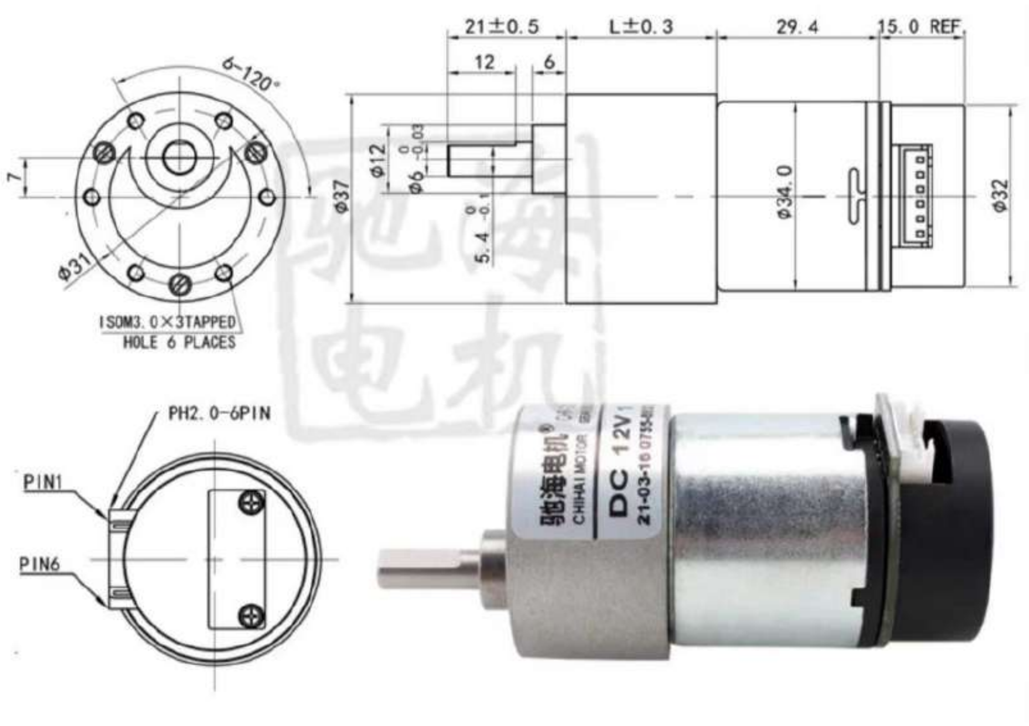


Figure 3: *Reducer motor and engineering drawing*

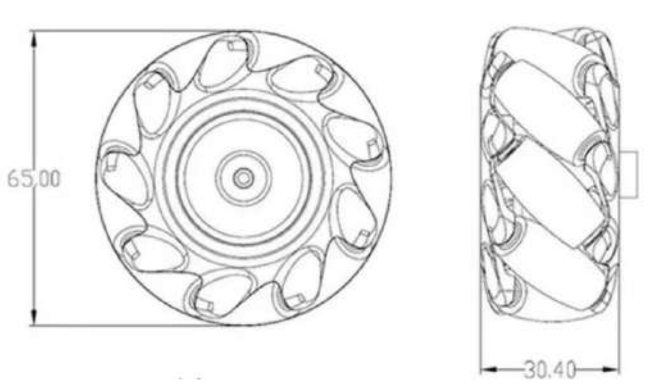


Figure 4: *Mecanum Wheel engineering drawing*

Figure 5: *Mecanum Wheel*

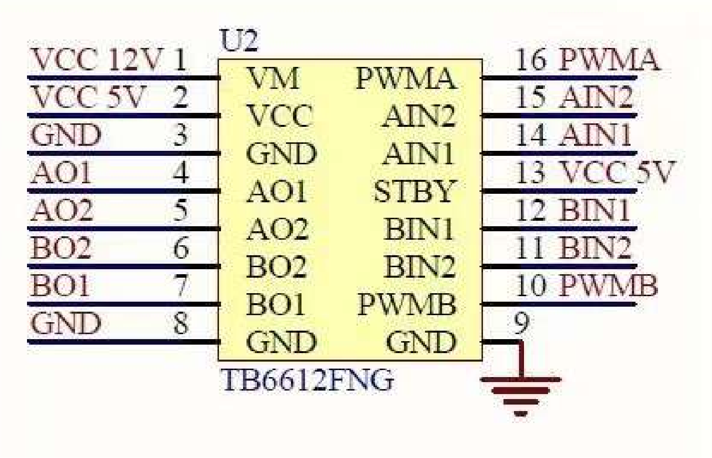


Figure 6: *Schematic diagram of the motor drive module*

2.1.3 Obstacle avoidance module

Today, with the continuous development of science and technology, there are more and more types of sensors that can be applied to smart cars[5]. Among many sensors, ultrasonic sensors have become the first choice for smart cars to achieve automatic obstacle avoidance applications. Ultrasound is a kind of mechanical wave with relatively strong directivity, high frequency and small diffraction phenomenon, so it is very suitable for ranging. The obstacle avoidance module used in this design is HC-SR04, which has 4 interface terminals, namely VCC, GND, receiving terminal Echo, control terminal Trig (Figure 7). It uses IO triggering for distance measurement. By using the high-level duration of the ultrasonic signal from the control end to the receiving end and the propagation principle of sound waves in the air, it can calculate the distance of obstacles and complete the function of obstacle avoidance (Figure 8). In addition, the normal working voltage of HC-SR04 is around 5 V, the measurement angle is less than or equal to 15ř, and the detection angle is larger when the distance is farther from the object.

In this design, the combination of HC-SR04 and SG90 steering gear is used to collect the position information of obstacles. When the smart car encounters an obstacle, the signal can be reflected to the ultrasonic sensor, and after analyzing and processing the position information of the obstacle, it is fed back to the Arduino processing chip, so that it can call the corresponding program to take countermeasures. According to the reflection effect of the ultrasonic wave and the rotation of the steering gear from 0ř to 180ř, the purpose of avoiding obstacles is realized[6].

Figure 7: *HC-SR04*

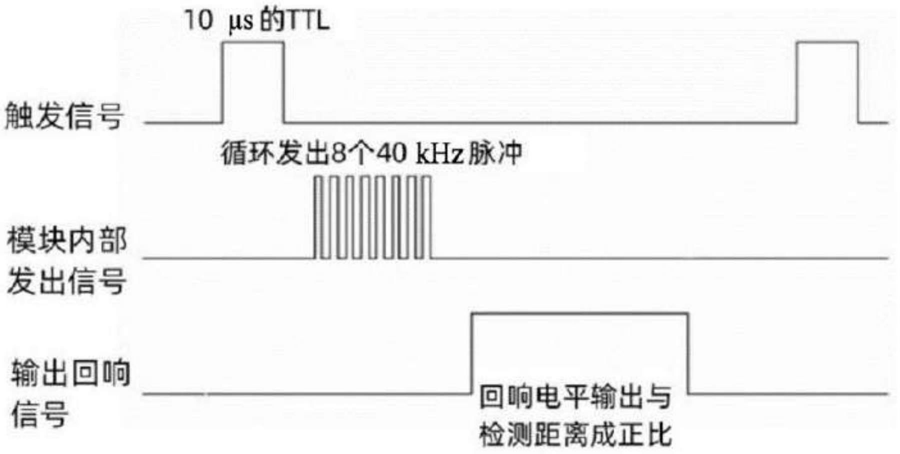


Figure 8: *Ultrasonic Timing Diagram*

2.1.4 Infrared tracking module

An 8-way anti-interference grayscale sensor is selected. Compared with the traditional sensor, this one can filter the external light source, and there is no need to re-adjust in different light environments. At the same time, the sensitivity adjustment function of the traditional sensor is retained, and the 3.3V level output is added, which is compatible with more microcontrollers. Signal output increases IIC and serial communication, and one bus can mount more modules (Figure 9, 10).

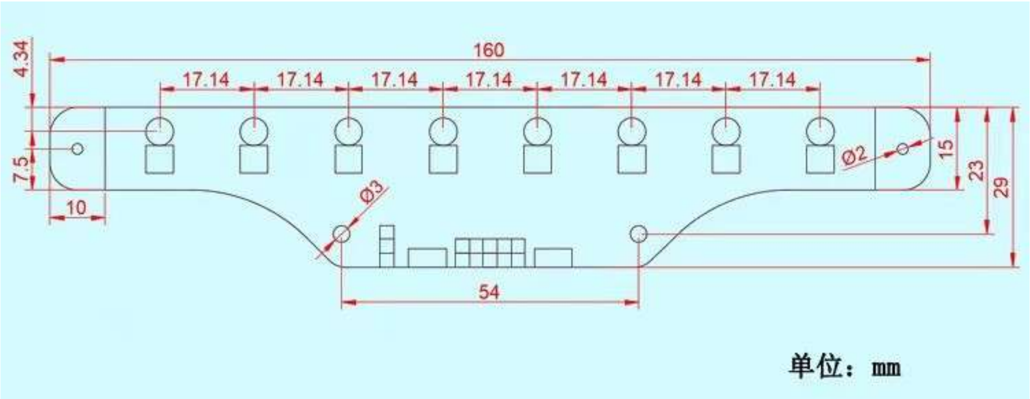


Figure 9: *8-way grayscale sensor engineering drawing*

Figure 10: *8-way grayscale sensor*

2.2 Manipulator system

In the overall design stage of the manipulator, the implementable scheme is proposed according to the use requirements. The design scheme is determined by measuring the available hardware and processing methods and comprehensively considering the analysis methods of kinematics, dynamics and mechanical design.

2.2.1 Design scheme of manipulator

As the skeleton part of the whole manipulator system, the manipulator frame plays a key role in installing the steering gear, placing open MV or other sensors, wiring and carrying the weight of the load block. Reasonably designing the mechanical structure of each joint can not only make the whole system look simple, but also better design can simplify the motion process, reduce the difficulty of the system in the subsequent parameter adjustment process, and enhance the stability and anti-interference of the system in the working process (Figure 11, 12).

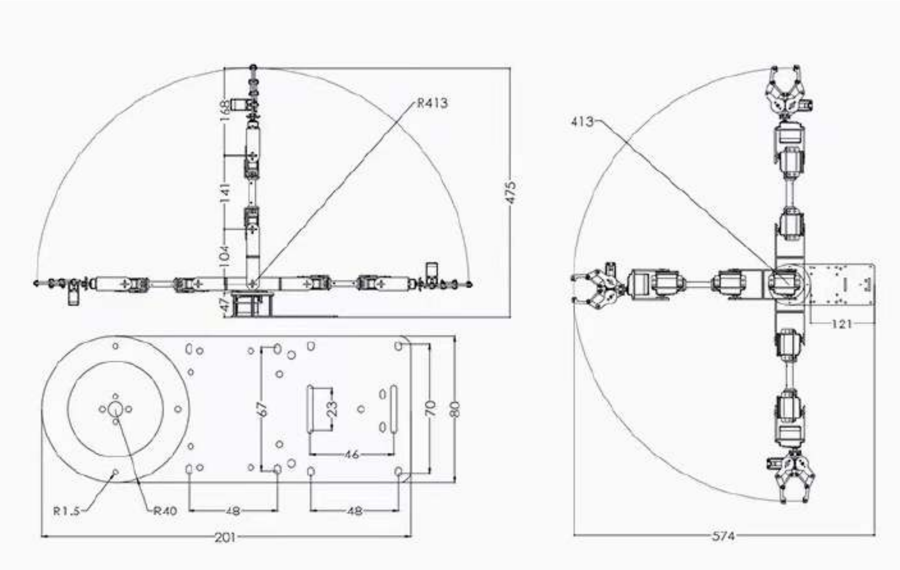


Figure 11: *6-DOF manipulator engineering drawing*

Figure 12: *6-DOF manipulator*

2.2.2 Hardware circuit design

The hardware circuit of the manipulator of the intelligent logistics car is mainly divided into three parts: motion execution unit, information processing unit and information acquisition unit. 6V DC power supply supplies power to all units (Figure 13).

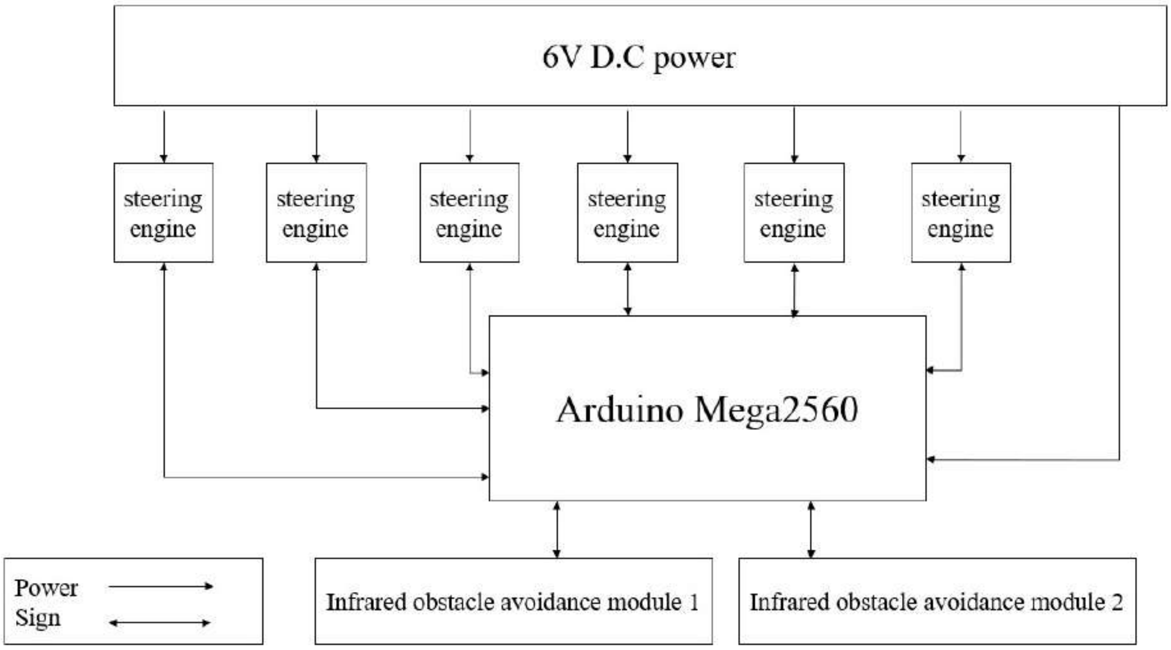


Figure 13: *Hardware circuit design*

▷ **Motion execution unit**

For how to drive the frame structure of the manipulator, after comprehensive consideration, we choose to use the steering gear YF61 as the actuator. The steering gear YF61 is an integrated servo unit with double bearings inside. It has the advantages of low friction loss, low noise, smooth operation, high-precision output, simple control and easy communication with single chip microcomputer. It is suitable for places where the angle needs to be changed or maintained. When the manipulator system grabs and places objects, the angle of each joint is changing all the time. Using this digital steering gear can meet the use requirements (Figure 14, 15, 16).

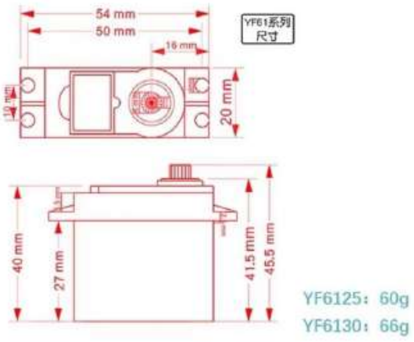
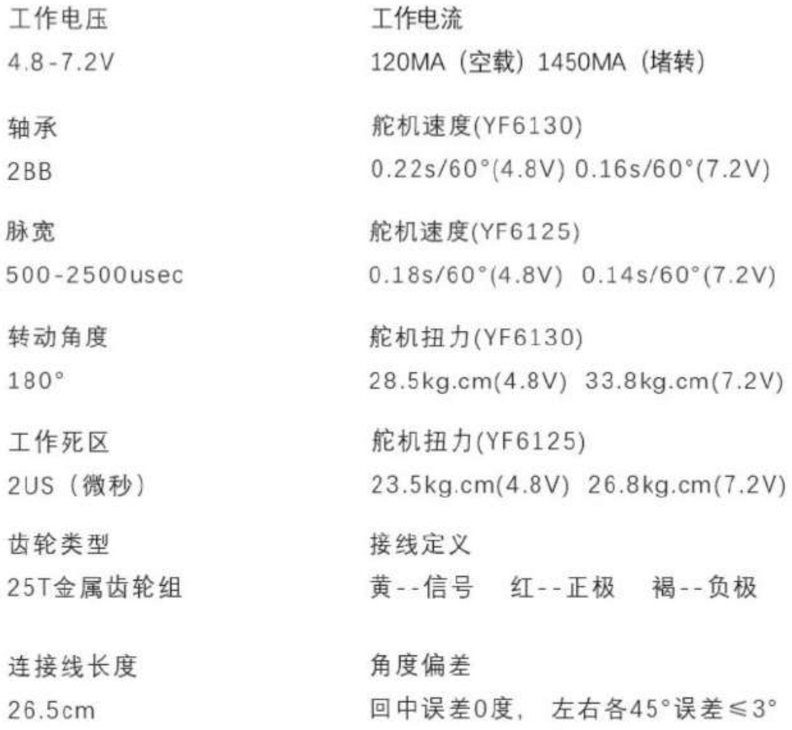


Figure 14: *Servo engineering drawing*

Figure 15: *Servo*



工作电压	4.8-7.2V	工作电流	120MA（空载）1450MA（堵转）
轴承	2BB	舵机速度(YF6130)	0.22s/60°(4.8V) 0.16s/60°(7.2V)
脉宽	500-2500usec	舵机速度(YF6125)	0.18s/60°(4.8V) 0.14s/60°(7.2V)
转动角度	180°	舵机扭力(YF6130)	28.5kg.cm(4.8V) 33.8kg.cm(7.2V)
工作死区	2US（微秒）	舵机扭力(YF6125)	23.5kg.cm(4.8V) 26.8kg.cm(7.2V)
齿轮类型	25T金属齿轮组	接线定义	黄--信号 红--正极 褐--负极
连接线长度	26.5cm	角度偏差	回中误差0度，左右各45°误差≤3°

Figure 16: *Servo parameters*

▷ **Information processing unit**

Arduino Mega2560 is used to send execution commands to the manipulator system. Because Arduino Mega2560 is the core circuit board with USB interface, it is very convenient for serial communication with computers, and has 14 digital I / O pins, which is suitable for the system design of steering gear, which needs a large number of IO interfaces (Figure 17).

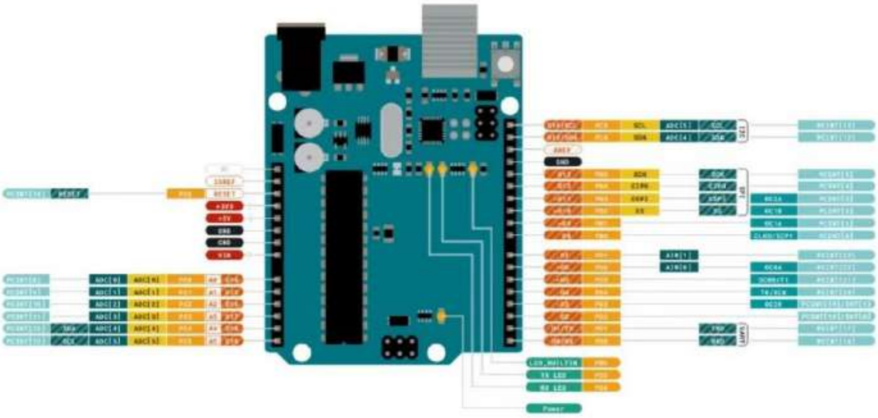


Figure 17: *Arduino Pin Diagram*

▹ **Information acquisition unit**

In this part, our group considered three different modules, debugged them respectively, and compared their advantages and disadvantages. The specific contents will be detailed in 3.24.

◊ **Visual module**

Color information is obtained through open MV. Open MV is an embedded image processing system. Its camera is a compact, low-power and low-cost circuit board. It can easily complete machine vision applications. Through the intelligent image recognition algorithm, open MV can quickly and accurately identify the color and position of the object and execute the corresponding instructions (Figure 18).

Figure 18: *Open MV*

◊ **Infrared obstacle avoidance module**

The detection distance of the sensor can be adjusted by potentiometer. It has the characteristics of small interference, easy assembly and convenient use. It can be widely used in many occasions, such as robot obstacle avoidance, obstacle avoidance car and black-and-white line tracking (Figure 19).

Figure 19: *HJ-IR2*

◊ **Ultrasonic module**

Compared with laser ranging, infrared ranging and other sensors, ultrasonic is insensitive to external light, color and electromagnetic field, and has strong environmental adaptability. In particular, it has great advantages in identifying transparent objects[7], especially the water bottle required to be grabbed by the project (Figure 20).

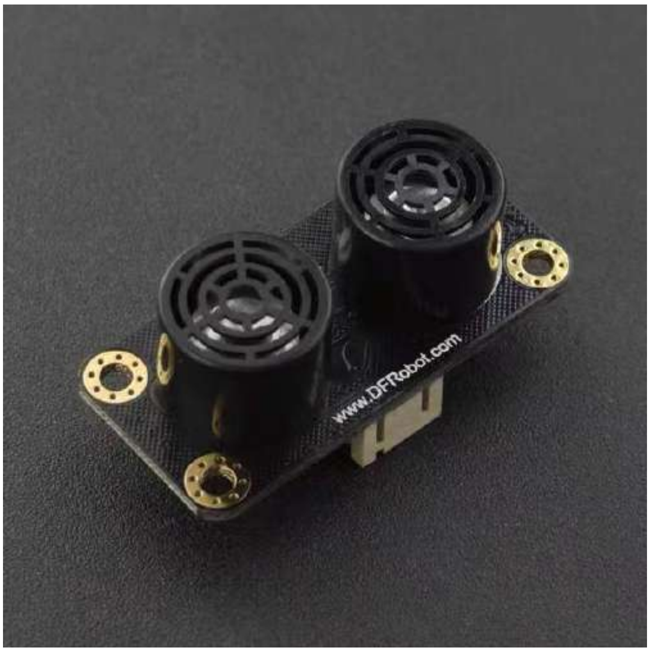


Figure 20: *DFRobot URM09*

3 Manufacturing

3.1 System principle

3.1.1 Principle of obstacle avoidance

The sensor designed for obstacle avoidance is HC-SR04, which uses IO-based triggering to realize the ranging function. During ranging, the Trig terminal will first send a pulse trigger signal of more than 10 s, and the module will cyclically send out 8 square wave pulse signals, the frequency of each pulse signal is 40 kHz, and it can judge whether there is an object in front of the sensor by automatically detecting whether there is an echo or not. If an echo is detected, the output signal in the Echo terminal is a high level, and the duration of the ultrasonic wave can be known from the time it takes for the ultrasonic wave to be sent to being received. The calculation formula is:

$$T = \frac{(Rh \times V_i)}{2} \tag{1}$$

T is the test distance;Rh is high level time;V_i is the speed of sound (340 m/s)

The application principle of HC-SR04 is that the high level should be sent by the control port first, and its duration is 10 s or more, and then detect the high-level output at the receiving port. Once the output signal is detected, the timer starts timing. After a period of time, if the signal in the receiving port changes from high level to low level, the value of the timer can be read. At this time, the value recorded by the timer is the ranging time. The measured distance can be calculated from the relationship between time, sound speed and distance.

3.1.2 Principle of tracking

We used an 8-way anti-interference grayscale sensor that returns an analog value. Each of these sensors can output analog values for black and white, ranging from 0 to 100. To be able to combine the values of the 8 sensors to get an offset to the black line, we use "QTRSensors" library in Arduino, through the library we can either calibrate the robot and then read the values of the sensors which depend on the initial values from the calibration or read the raw values of the sensors. The first case will be more accurate. After calibration, the sensor array will return the position of the black line. It can be between 0 and 700. If the position is 350, it means the sensor array is on the center of the line, then we subtract 350 from this value to get a value in the range (-350,350), which indicates the offset to the black line and can be used to be the error of each loop while tracking.

3.1.3 Kinematics analysis of manipulator

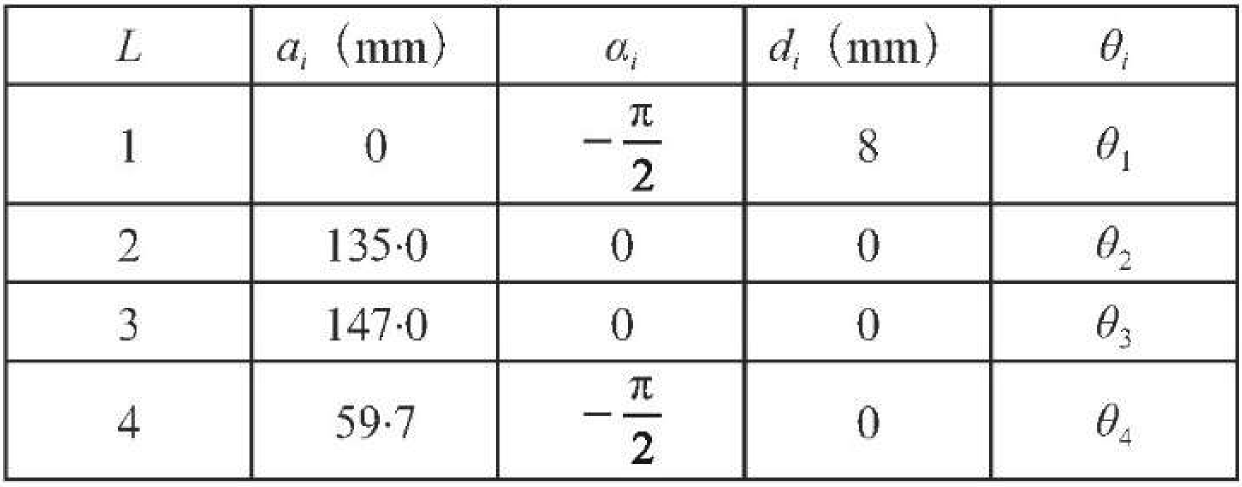
Robot simulation technology is of great significance in the design and research of 6-DOF manipulators. The Denavit-Hartenberg model, referred to as the D-H model, is a very simple modeling method proposed by Denavit and Hartenberg in 1955 for robotic systems. This modeling method is suitable for robots of various shapes and has long been a standard method for modeling in robot simulation technology.

Through the D-H model, the kinematics analysis of the manipulator can be divided into forward kinematics analysis and inverse kinematics analysis. This article gives a general overview of this part, and the details can be found in the relevant literature[8−13]. D-H modeling is a standard method of modeling robot kinematics, establishing a coordinate system on each link of the manipulator. By determining all the homogeneous coordinate transformations between each joint and the next joint, the attitude relationship between the end of the manipulator and the base is calculated. The homogeneous coordinate system uses n+1-dimensional vectors to represent n-dimensional vectors, and unifies the rotation transformation and translation transformation into a 4×4 matrix operation. The homogeneous transformation matrix is represented as follows:

$$ {}_i^jT = \begin{bmatrix} {}_i^jR & {}_i^{oj}P \\ 000 & 1 \end{bmatrix}_{4\times4} \tag{2} $$

${}_i^jR$ represents rotation, the right column represents translation.

There are four main parameters involved in the D-H modeling process: a is the length of the vertical line, α is the torsion angle between two adjacent z axes, d is the distance between the adjacent common perpendiculars of the z axis, is the joint rotation angle. The D-H parameter table of the manipulator is shown in Figure 21 [14].



L	a_i (mm)	α_i	d_i (mm)	θ_i
1	0	$-\frac{\pi}{2}$	8	θ_1
2	135·0	0	0	θ_2
3	147·0	0	0	θ_3
4	59·7	$-\frac{\pi}{2}$	0	θ_4

Figure 21: *D-H modeling parameters of the manipulator [14]*

Each row of the D-H parameter table corresponds to the homogeneous transformation matrix of each step. Multiplying these transformation matrices to the right can establish the coordinate relationship between the base and the end of the manipulator as shown in the following formula. By solving the joint variables, the control of the manipulator can be realized.

$$ {}_0^4T ={}_0^1 T_1^2T_2^3T_3^4T \tag{3} $$

In the automatic grasping system, we actually control the manipulator to reach the specified position when the target position is known, that is, we need to solve the value of all joint variables according to the position information, which is to solve its inverse kinematics equation.

3.2 System implementation

3.2.1 Obstacle avoidance scheme design

Through mathematical modeling, we analyze the path of the vehicle on a given track, and achieve the set goal through the embedded program debugging of the control algorithm. There are two main obstacle avoidance schemes in this paper.

▷ **Plan A**

As shown in Figure 22, from the starting line, drive in a straight line. After encountering an obstacle, use the ranging sensor to determine the direction with a larger distance, use the differential to turn, and then drive in a straight line; after encountering the next obstacle, repeat the above process. If the car encounters an area without obstacles, use the distance measuring sensors on both sides to determine the neutral position, and keep driving straight at the maximum speed.

Advantages: simple thinking, easy for overall analysis and design.

Disadvantages: The travel path is long, the time is long, and it is difficult to use the differential turning method to accurately turn 90ř in the programming of the program algorithm.

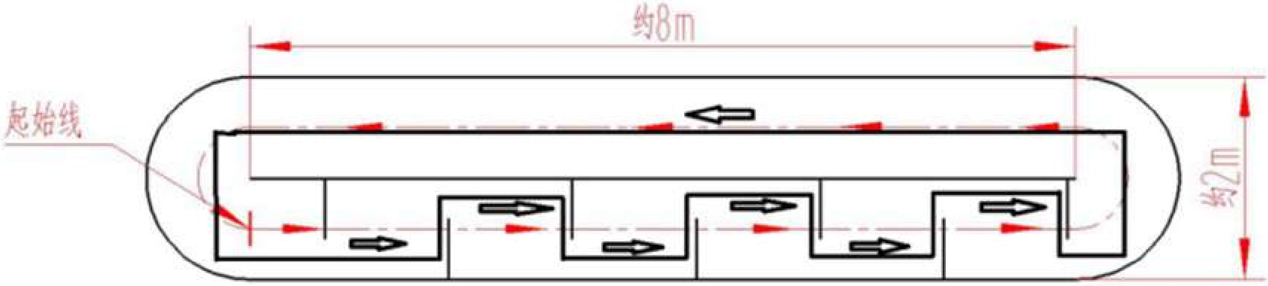


Figure 22: *Plan A*

▷ **Plan B**

As shown in Figure 23, Starting from the starting line, the ranging sensor measures the distance in real time, and the controller selects the widest space for steering in real time. Since the direction changes in real time, the simulated effect graph is a curve. Drive along the curve in the figure to avoid obstacles, and drive in a straight line after passing through the obstacle area.

Advantages: Compared with Plan A, the path is shorter, the time is less, and the program algorithm is convenient for the overall design.

Disadvantages: The driving correction link and the procedure is complicated.

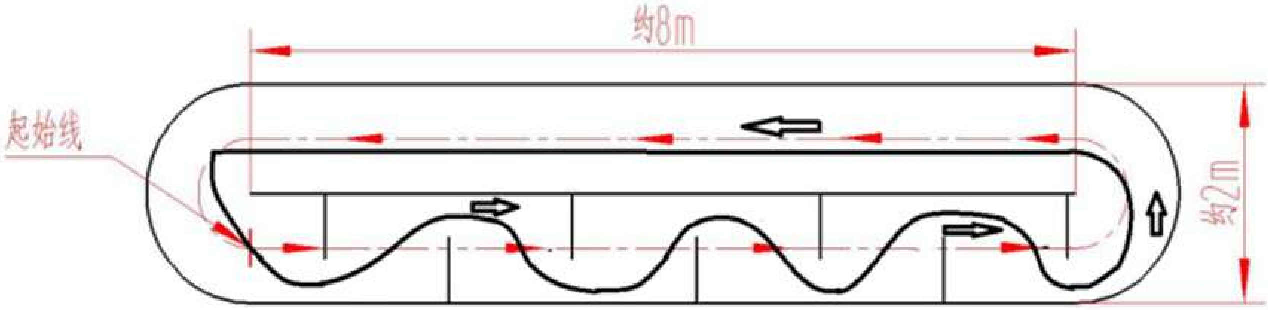


Figure 23: *Plan B*

In order to simplify the procedure, we choose Plan A. The scheme to solve the problem of turning 90ř is to use the Mecanum wheel. At the same time, the distance to avoid obstacles in the project is short, so the time cost can be ignored.

3.2.2 Tracking Algorithm and PID control

The distinguishing feature of the PID controller is the ability to use the three control terms of proportional, integral and derivative influence on the controller output to apply accurate and optimal control. For each term, it corresponds to a constant, Kp, Ki and Kd, that must be adjusted so that the robot can follow a line without oscillating or slowing down or getting off the track.

The proportional term is the error. It directly controls how to take the curves - if Kp is a small value it will take the curves easier (it will go almost straight); if it is a large value it will take the curves suddenly (either it will oscillate on a straight line, or it will take the curve too tight and it will leave the track).

The integral term accumulates all errors. The integral term seeks to eliminate the residual error by adding a control effect due to the historic cumulative value of the error. When the error is eliminated, the integral term will cease to grow. This will result in the proportional effect diminishing as the error decreases, but this is compensated for by the growing integral effect. In other words, it helps the robot stop oscillating. But at a Ki that is too high, it will do the opposite.

The derivative term calculates the current error and the last error. When the robot suddenly hits a tight curve, this value will be high and will force the robot to take the required curve. The more rapid the change, the greater the controlling or dampening effect. At a Kd too small, this value might not take place. At a Kd too high, it can give errors to the whole program and the robot can oscillate, run very slowly or take very narrow curves that don't even exist.

The whole point of this algorithm is finding the 3 constants. For our intelligent car Kp is 0.41, Ki is 0.0041and Kd is 2. We can change their values every time in the program, or put a Bluetooth module in which we can control these values directly from the phone (Figure 24-25).

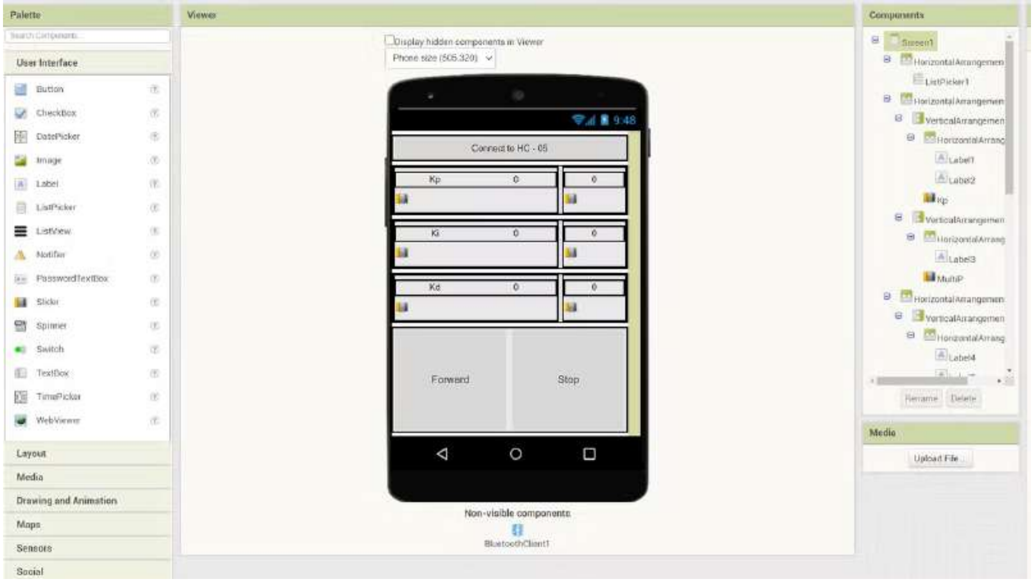


Figure 24: *MIT APP Inventor 1*

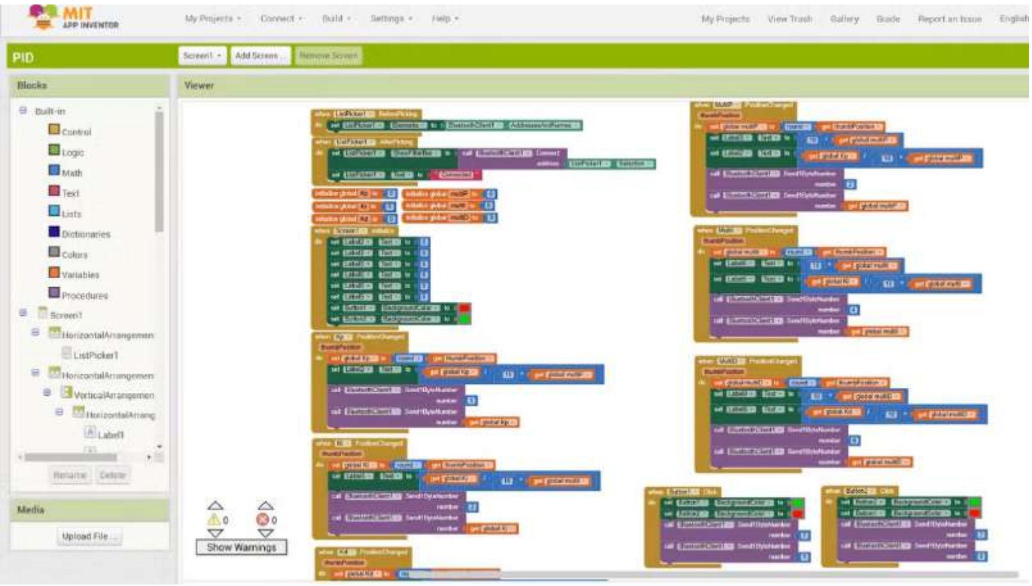


Figure 25: *MIT APP Inventor 2*

3.2.3 Manipulator system parameter adjustment

The manipulator is driven by six steering gears (numbered A, B, C, D, E, F). Among them, the F servo controls the opening and closing of the mechanical claw, and the E servo controls the rotation of the mechanical claw. Therefore, it is the remaining four steering gears that actually control the motion trajectory of the manipulator. The A steering gear controls the overall rotation of the manipulator, and the B, C, and D steering gears determine a plane and control the movement of the manipulator in this plane. Cylindrical space coordinates can thus be used (Figure 26).

In the plane determined by the B, C, and D steering gears, according to the design situation, determine the starting point position, ending point position and motion trajectory of the gripping point of the manipulator, as well as the number of rotation angles of each steering gear corresponding to the starting point and the ending point. The fixed point position can be determined by actual measurement. The determination of the steering gear angle is first obtained by theoretical calculation to obtain the estimated value, and then adjusted by the test method.

After the fixed point work is completed, the motion trajectory of the manipulator needs to be determined. Each steering gear between the start point and the end point rotates in the corresponding direction and angle, and the angle is added and subtracted in a linear proportion. If there are certain requirements for the trajectory, in order to improve the degree of trajectory fitting, the trajectory can be segmented. Determine the starting point, ending point position and steering angle of each track, and then add and subtract the angle in a linear proportion to realize the fitting of the motion track. The more segments, the shorter the length of each segment, and the higher the fitting degree. Since the analog servo cannot achieve complete synchronous rotation, the adjustment of the trajectory points and the fitting of the trajectory can only be carried out by reducing the single-step rotation angle and performing interpolation operations at the same time.

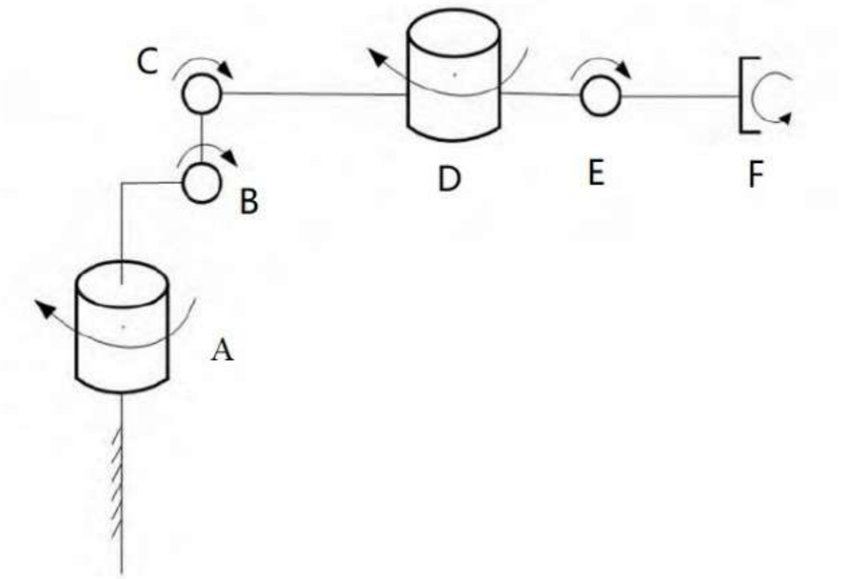


Figure 26: *Six steering gears of manipulator*

3.2.4 Grabbing object scheme design

▷ **Color Identification Scheme**

Machine vision algorithms on Open MV include finding color patches, face detection, eye tracking, edge detection, sign tracking, and more. This system mainly uses the algorithm for finding color blocks and the algorithm for finding color rings. The first is to find the largest color block algorithm. There are many impurities in the background. In order to reduce the influence of impurities, a noise reduction algorithm is also added, which can make the identification of color blocks more accurate. The threshold of color block color adopts Lab, Lab color space, L represents brightness, positive value of A represents red, negative value A represents green; positive value of B represents yellow, and negative value B represents blue. Unlike RGB and CMYK color spaces, Lab colors are designed to approximate human vision. Set the threshold structure of a color as (min L, max L, min A, max A, min B, max B). After judging whether the color block is the specified block, Open MV and Arduino communicate through strings to realize the grasping of the mechanical claw or the next position of the manipulator. When the manipulator receives the signal to put down the block, it first judges whether it is the correct color through Open MV. If it is correct, it will mark the center coordinates and send it to Arduino, and then the Arduino will control the manipulator to place it.

Shortcoming: During the test, it was found that Open MV has a big drawback that it is extremely sensitive to changes in light, and changes in light intensity have a huge impact on the recognition accuracy of color blocks. Therefore, in subsequent improvements, Open CV will be used instead of Open MV for color patch recognition. Open CV has a wealth of algorithms commonly used in image processing and computer vision, and supports machine learning and deep learning. The machine learning library focuses on statistical pattern recognition and clustering, and the deep learning library focuses on vision tasks.

In subsequent improvements, the Tensorflow model in Open CV will be used to analyze and train the color and shape of the color blocks. Taking pictures multiple times, long-term learning can better improve the training results. After using Open CV, it can ensure that the influence of light on the color block recognition is greatly reduced, and the recognition will be more accurate and rapid.

▷ **Ultrasonic positioning solution**

The working principle of ultrasonic sensor ranging is that after the ultrasonic wave is sent out, it propagates in the air at a speed v, and is reflected back when it reaches the surface of the detected object, and is received by the ultrasonic transmitter. The round-trip time is t, then the distance S is measured. The formula is:

$$S = \frac{vt}{2} \tag{4}$$

After the ultrasonic sensor measures the distance to the object, it converts the measured data into binary signals that Arduino can recognize. After processing the binary signals, Arduino sends commands to the steering gear, and then the steering gear drives the manipulator to move.

Shortcoming: The ranging speed is slow, resulting in a large error in each grab; if the surface of the object to be grabbed is flat, the effect is better, and if it is an irregular surface or a curved surface, the error is large. At the same time, the ultrasonic module has been used in the obstacle avoidance process of the car. In order to differentiate, we decide not to use the second ultrasonic module.

▷ **Infrared obstacle avoidance positioning**

In order to enable the cart to stop precisely when it recognizes an object, we use high-precision infrared obstacle avoidance modules, one in the front and the other in the middle of the cart. When the front module recognizes an object, the car enters low-speed trajectory mode and stops when the back one recognizes the object, passing the pinch signal to the manipulator.

Shortcoming: Initially, the car didn’t always stop immediately when it recognized an object, it would continue to move forward a bit due to inertia.

To solve this problem, we also use encoders, and use the encoders to implement PID control on the motors so that the car could reduce the speed to zero as quickly as possible when it receives a stop signal.

Finally, through the analysis of the above scheme, we decide to use the infrared obstacle avoidance module to locate the position of the object

3.2.5 Dumping device

We designed a dumping device consisting of a steering gear and a box-like device. During the process of the intelligent car driving and grasping objects, the dumping device is in a stable state. When the car reaches the dumping area, the steering gear will control the dumping device to lift up, so that the object is dumped into the area.

3.3 System validation

3.3.1 Obstacle avoidance

In order to verify the obstacle avoidance performance of the car, some simple objects are used as obstacles to specify the specific route of the car when setting up the experimental environment, so as to verify the obstacle avoidance effect of the car.

The intelligent car detects the surrounding environment through ultrasonic sensors during driving. The ultrasonic sensor can continuously detect whether there are obstacles in the left, front and right directions. In case of obstacles, the car will compare the distance of obstacles according to these three directions, so as to make driving judgment to avoid obstacles automatically. The car may encounter obstacles in different directions, generally left, right and front. In order to take effective measures to avoid obstacles, it is necessary to develop obstacle avoidance algorithms in these three directions to accurately judge the position information of obstacles (Figure 27).

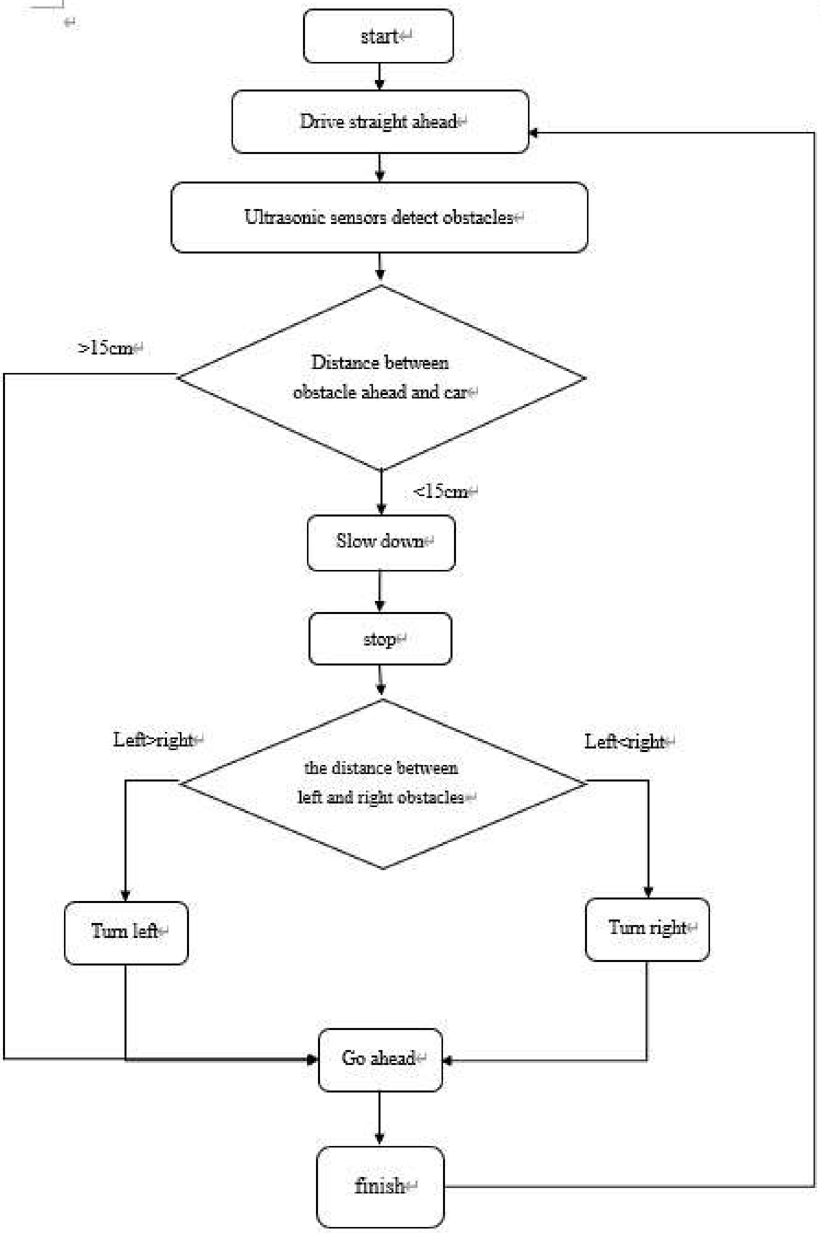


Figure 27: *Flow chart of intelligent car obstacle avoidance*

The ultrasonic sensor first detects the distance between the objects directly in front. If there is no object more than 15 cm in front of the car, the car will continue to drive forward. If an object within 15 cm ahead is detected, the car will decelerate, and the HC-SR04 sensor will detect the distance between the left and right obstacles at the same time. If the distance between the left obstacles is greater than the right, the car will turn left and then move forward. If the distance between the obstacles on the right is greater than that on the left, the car will turn right and then move forward. If there are obstacles in front of the car, the car will repeat the above process and adjust the angle to avoid all obstacles. The experimental results are shown in Figure 28.

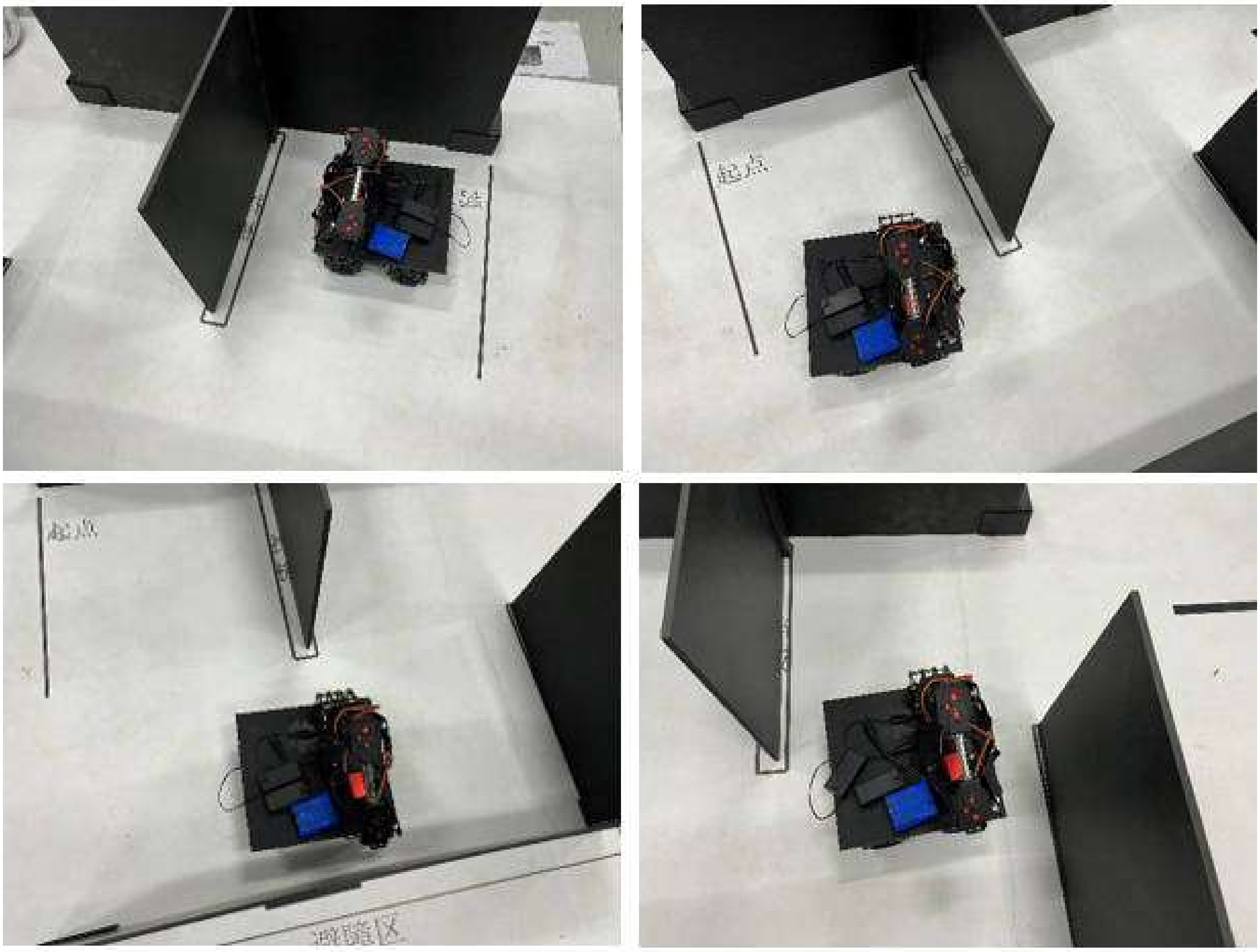


Figure 28: *Obstacle avoidance experiment of intelligent car*

3.3.2 Tracking

In order to verify the tracking performance of the car, when setting the experimental environment, the teacher designed curves, right angle turns and continuous turns as specific routes for the car to verify the tracking effect of the car. When the smart car is moving forward, the infrared sensor constantly detects the front. When white is detected, the infrared sensor judges the position of the car and returns a signal. Arduino calls the program to control the middle position of the car to always remain on the black line according to the feedback signal, so as to effectively drive along the black line. The experimental effect is shown in Figure 29.

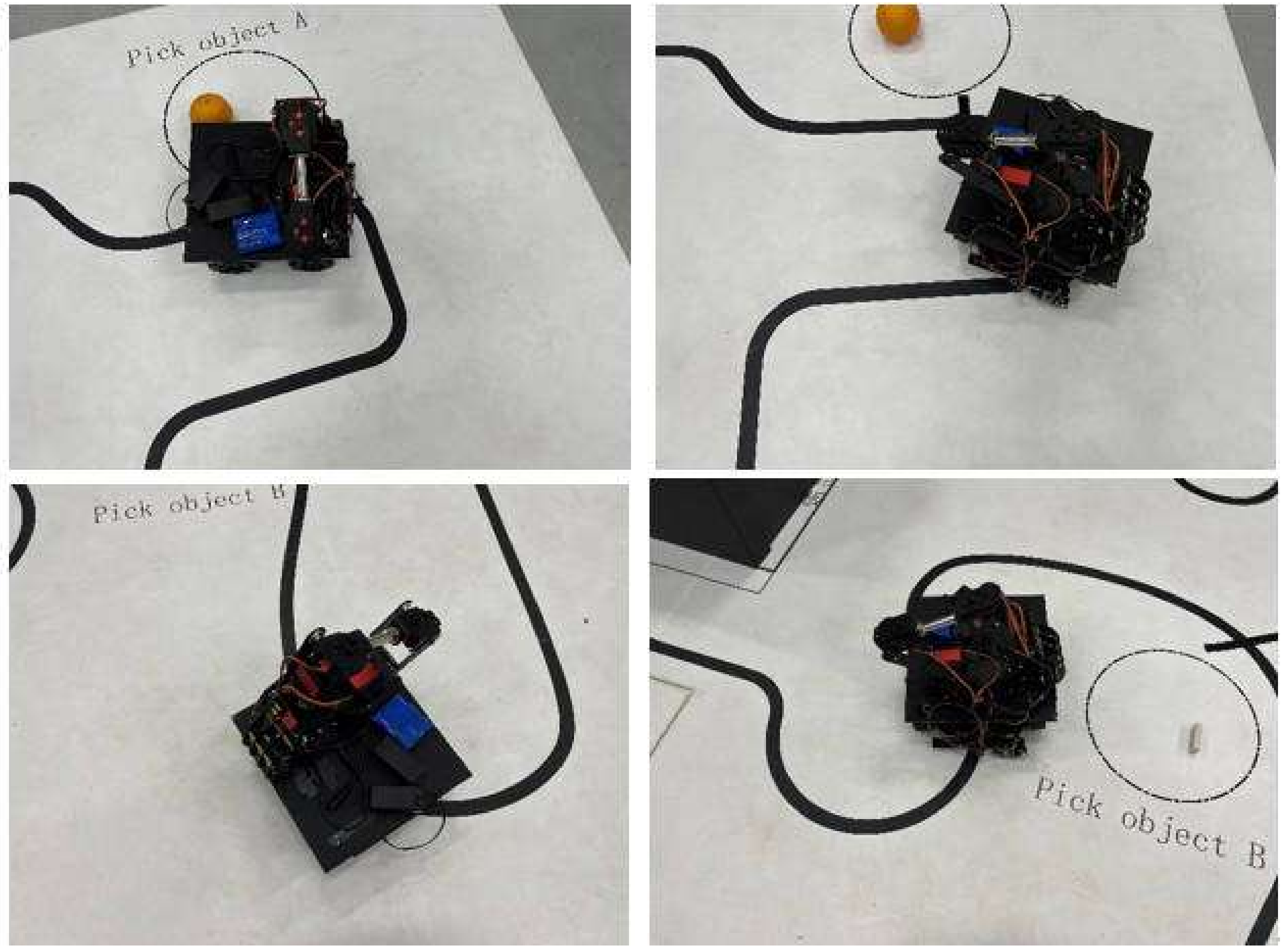


Figure 29: *Tracking experiment of intelligent car*

3.3.3 Grabbing

The initial position of the recorded object is the theoretical position. Adjust the intelligent car to the initial state, and start the system for Grab Test. When the manipulator runs to the grasping position and starts the grasping operation, record that the current grasping position of the manipulator is the actual position. The distance between the theoretical position and the actual position is recorded as an error. See Table 1 for details. After calculation, the average error of 10 experimental samples is 1.935 mm.

Table 1: *Grab position error*

Number	Error(mm)	Number	Error(mm)
1	1.58	6	2.41
2	1.81	7	1.52
3	2.33	8	1.94
4	2.07	9	2.15
5	1.92	10	1.70

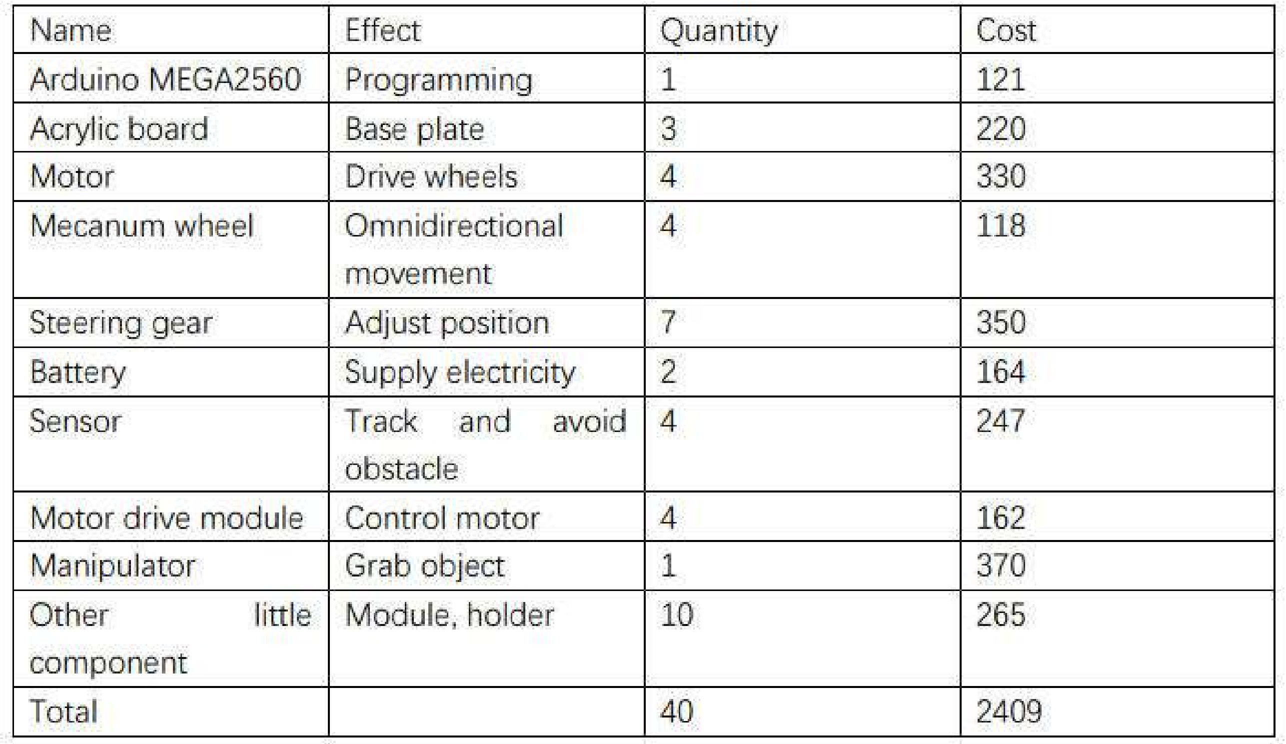
3.3.4 Unloading

The center of the unloading area is the theoretical position. Adjust the smart car to the initial state and start the system for unloading test. When the unloading device runs to the unloading position and starts the unloading operation, the current unloading position of the unloading device is recorded as the actual position. The distance between the theoretical position and the actual position is recorded as the error. See Table 2 for details. After calculation, the average error of 10 unloading experimental samples is 2.12 mm

Table 2: *Unload position error*

Number	Error(mm)	Number	Error(mm)
1	1.58	6	2.41
2	1.81	7	1.52
3	2.33	8	1.94
4	2.07	9	2.15
5	1.92	10	1.70

4 Cost Estimation

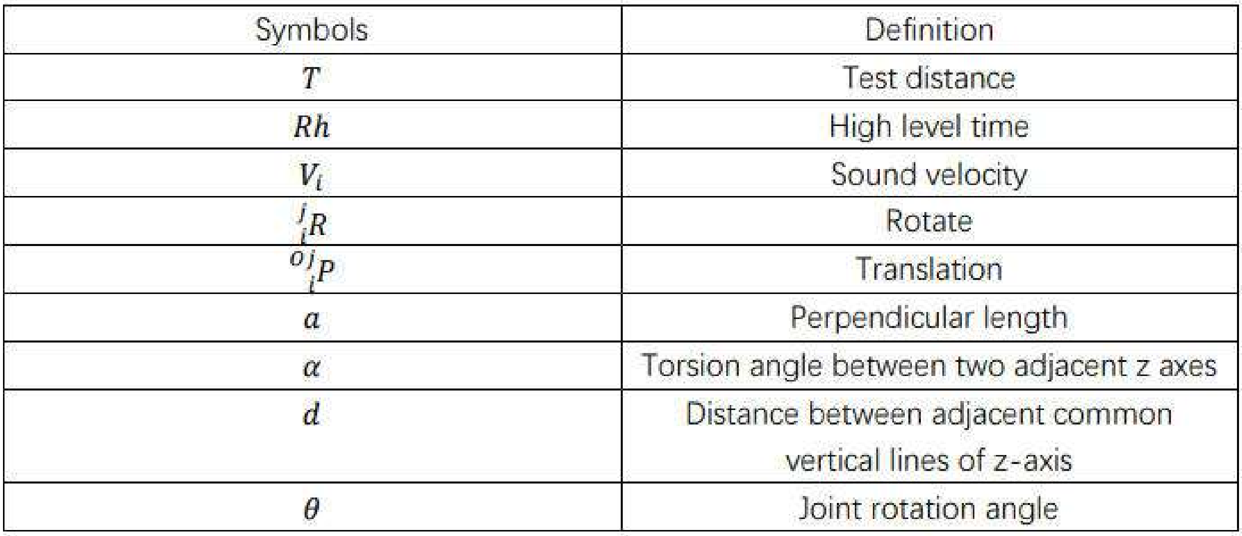


Name	Effect	Quantity	Cost
Arduino MEGA2560	Programming	1	121
Acrylic board	Base plate	3	220
Motor	Drive wheels	4	330
Mecanum wheel	Omnidirectional movement	4	118
Steering gear	Adjust position	7	350
Battery	Supply electricity	2	164
Sensor	Track and avoid obstacle	4	247
Motor drive module	Control motor	4	162
Manipulator	Grab object	1	370
Other little component	Module, holder	10	265
Total		40	2409

5 Conclusion

This design uses Arduino processing chip, combined with power supply, motor drive, steering gear, ultrasonic and other modules to develop an automatic obstacle avoidance car, and verifies the feasibility of the car's automatic obstacle avoidance. By designing the car to run along the specified obstacle avoidance route, the test shows that the car has good obstacle avoidance performance. At the same time, the process of manipulator automatically grasping and placing different objects is also very successful. However, the smart car still has some shortcomings. For example, the car can only automatically detect the obstacles in front, left and right, but can not detect the obstacles in the height direction, or it fails to successfully grab the mineral water bottle with open MV, resulting in having to replace it with other sensors. If it is replaced with other algorithms, the effect may be greatly improved.

6 Nomenclature



Symbols	Definition
T	Test distance
Rh	High level time
V_i	Sound velocity
$^{j}_{i}R$	Rotate
$^{oj}_{i}P$	Translation
a	Perpendicular length
α	Torsion angle between two adjacent z axes
d	Distance between adjacent common vertical lines of z-axis
θ	Joint rotation angle

7 Acknowledgements

Thank the school for providing us with experimental sites, experimental facilities and project funds. Thanks to the teachers, teaching assistants and warm-hearted students who gave us help and advice in the process of the project.

References

[1] Robin C, Lacroix S. Multi-robot target detection and tracking: taxonomy and survey[J]. Autonomous Robots, 2016, 40(4): 729-760.

[2] Pereira V, Fernandes V A, Sequeira J. Low cost object sorting robotic arm using Raspberry Pi[C]//2014 IEEE global humanitarian technology conference-South Asia Satellite (GHTC-SAS). IEEE, 2014: 1-6.

[3] Badamasi Y A. The working principle of an Arduino[C]//2014 11th international conference on electronics, computer and computation (ICECCO). IEEE, 2014: 1-4.

[4] Alli K S, Onibonoje M O, Oluwole A S, et al. Development of an Arduino-based obstacle avoidance robotic system for an unmanned vehicle[J]. ARPN Journal of Engineering and Applied Sciences, 2018, 13(3): 1-7.

[5] Mu F, Liu C. Design and Research of Intelligent Logistics Robot based on STM32[J]. Recent Advances in Electrical & Electronic Engineering (Formerly Recent Patents on Electrical & Electronic Engineering), 2021, 14(1): 44-51.

[6] Jin Y, Li S, Li J, et al. Design of an intelligent active obstacle avoidance car based on rotating ultrasonic sensors[C]//2018 IEEE 8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER). IEEE, 2018: 753-757.

[7] Bhargava A, Kumar A. Arduino controlled robotic arm[C]//2017 International conference of Electronics, Communication and Aerospace Technology (ICECA). IEEE, 2017, 2: 376-380.

[8] Sun Lele. Simulation Analysis of 6-DOF Manipulator[J]. Journal of Physics: Conference Series, 2021, 2033(1)

[9] Li Xianglong and Quan Zikun and Liu Dongping. Design of Control System for 6-DOF Manipulator[J]. IOP Conference Series: Materials Science and Engineering, 2020, 772(1) : 012041.

[10] Junhao Zhang et al. Path Planning Simulation of 6-DOF Manipulator[J]. Journal of Physics Conference Series, 2020, 1574(1) : 012156.

[11] Robotics; Researchers from University of Tokyo Discuss Findings in Robotics (Working Environment Design for Effective Palletizing with a 6-DOF Manipulator)[J]. Journal of Robotics & Machine Learning, 2016,

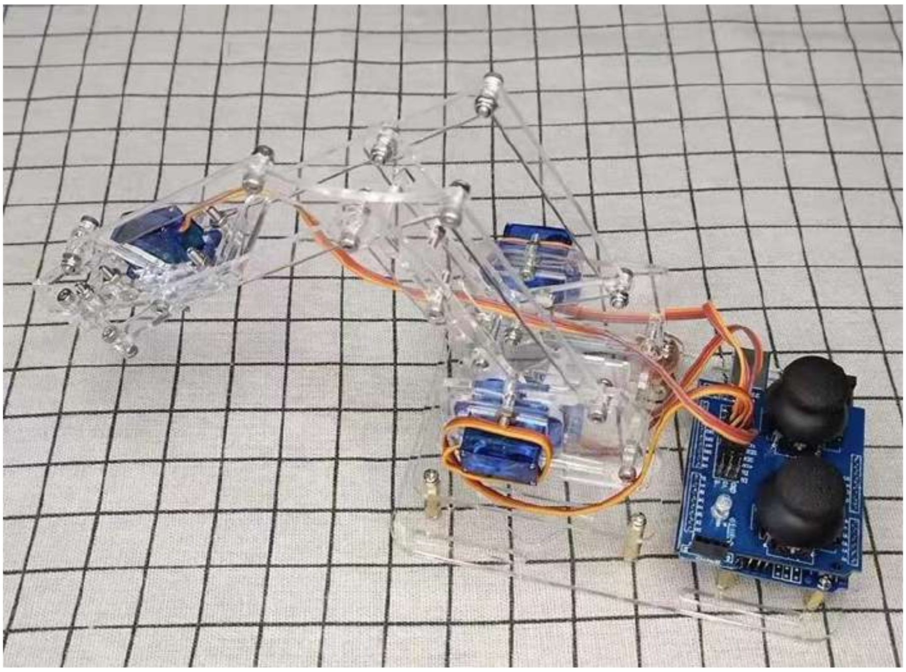
[12] Wei Hua Su et al. Task-Oriented Servo Loops Control of a 6-DOF Manipulator for Rescue Robot[J]. Applied Mechanics and Materials, 2013, 2667(404-404) : 663-667.

[13] Jacques A. Gangloff and Michel F. de Mathelin. Visual servoing of a 6-DOF manipulator for unknown 3-D profile following[J]. IEEE Transactions on Robotics and Automation, 2002, 18(4) : 511-520.

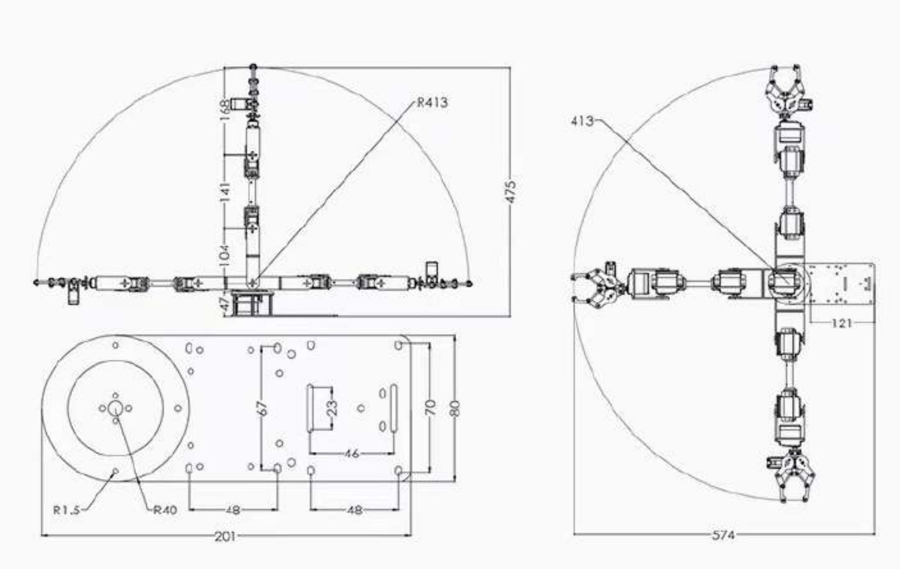
[14] Kofman J, Wu X, Luu T J, et al. Teleoperation of a robot manipulator using a vision-based human-robot interface[J]. IEEE transactions on industrial electronics, 2005, 52(5): 1206-1219.

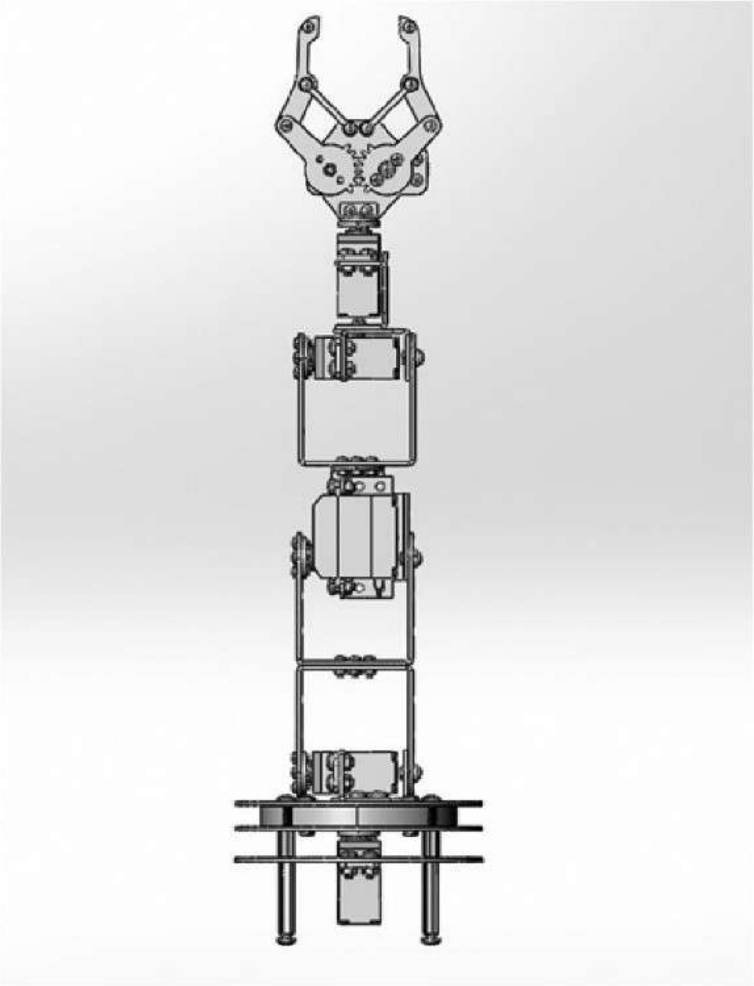
Appendix

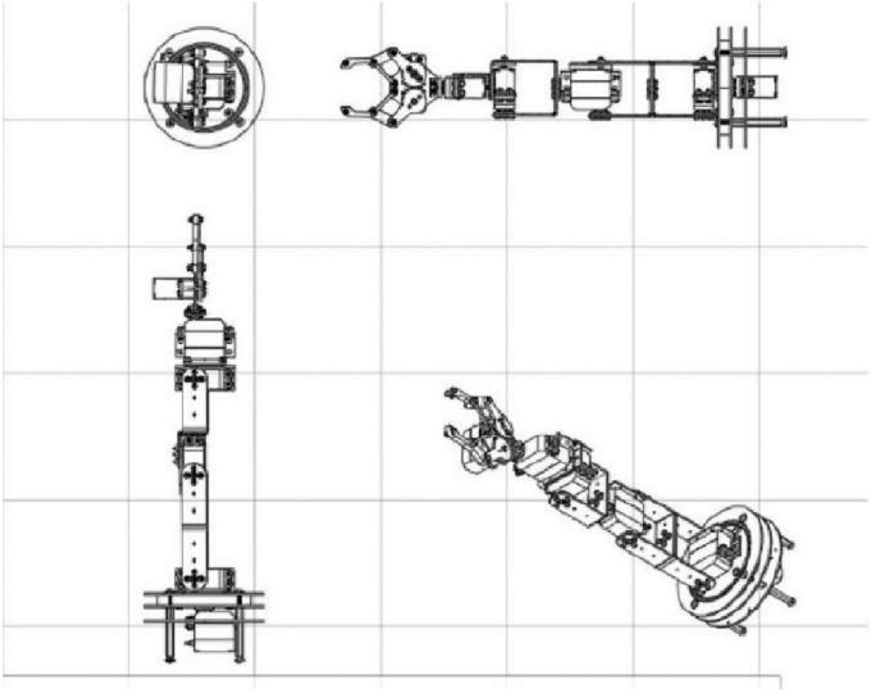
A All sketches in concept design

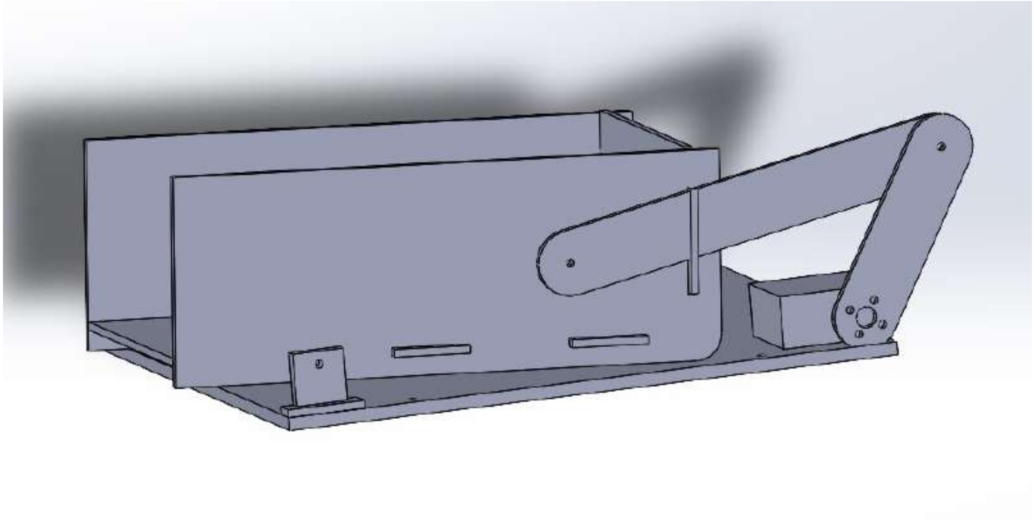


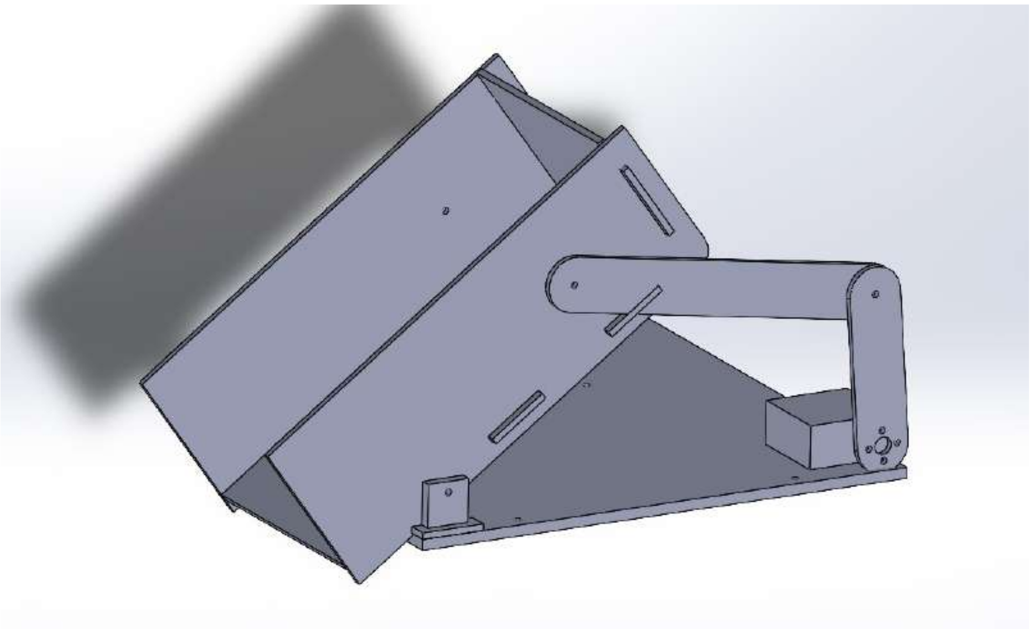
B All engineering draws with SolidWorks









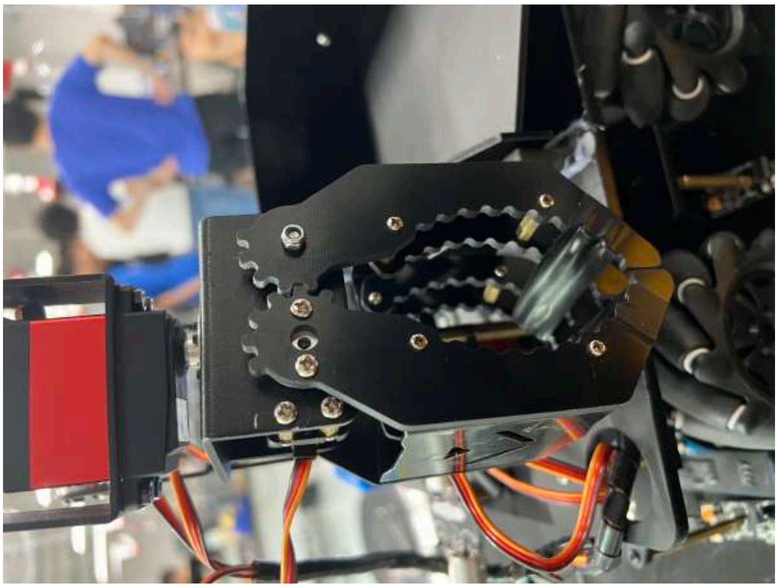


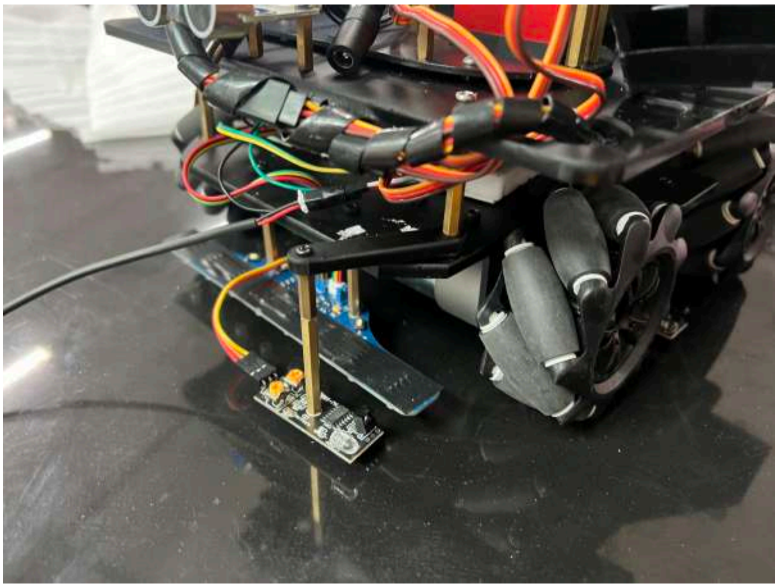
C Product Design Specifications

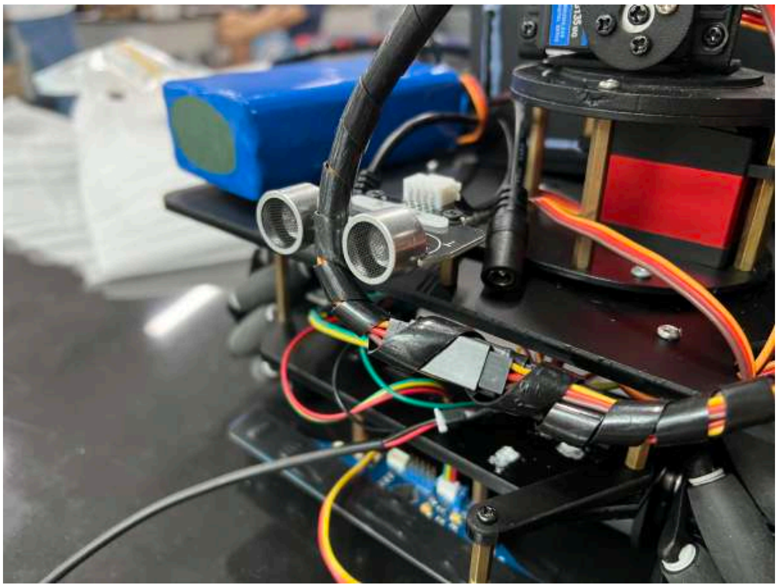
Table 3: *Product Design Specifications*

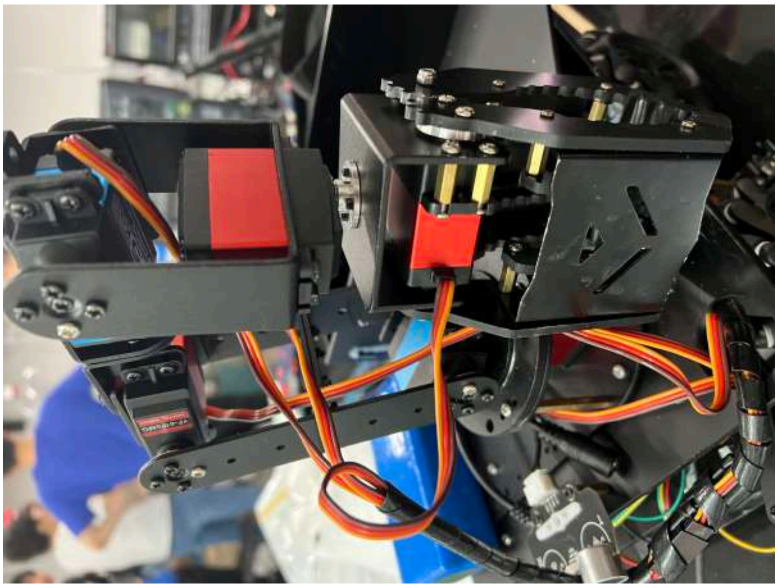
Product Design Specifications
Product Identification
Name: Intelligent car
Functions: Tracking, obstacle avoidance, automatically grabbing objects
Special Features: Stable performance in obstacle avoidance, flexible manipulator, Mecanum wheels, Automatic control
Service environment: Normal temperature and pressure, light under any conditions

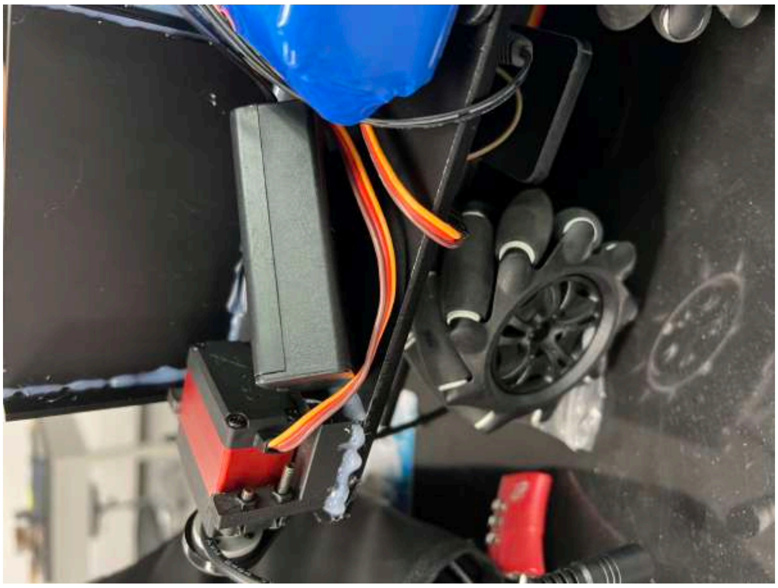
D Details of Prototyped machines

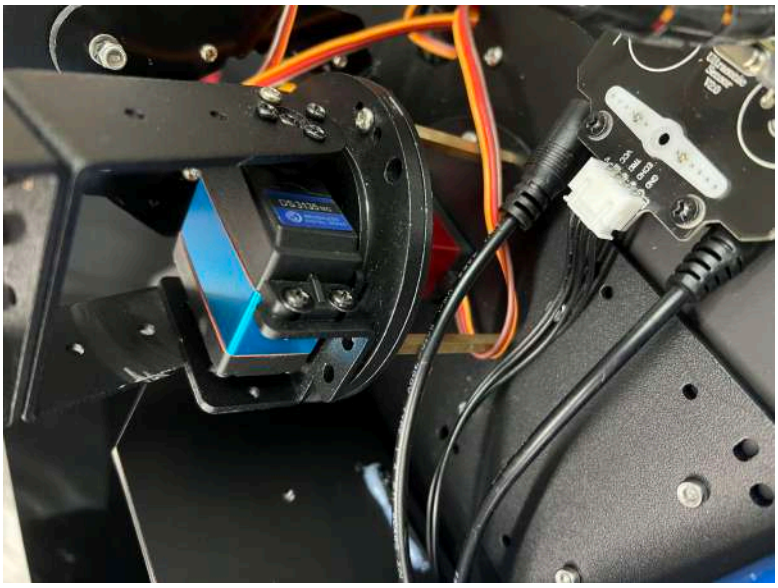


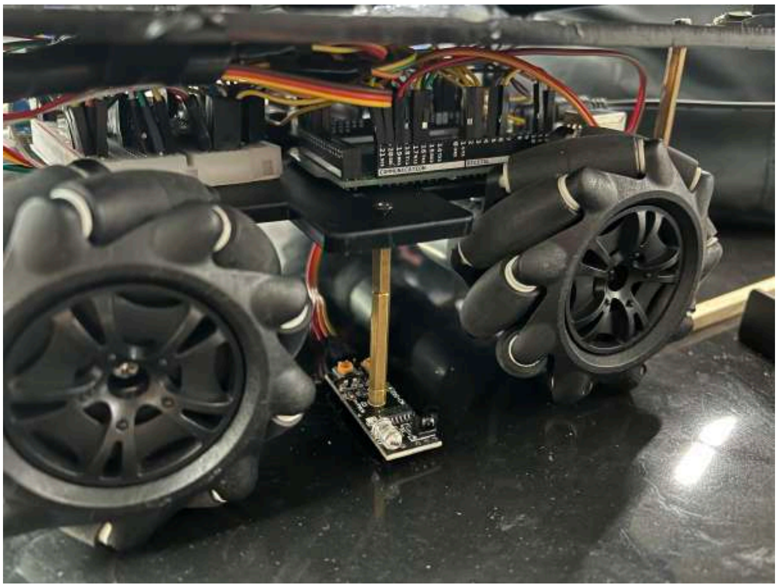


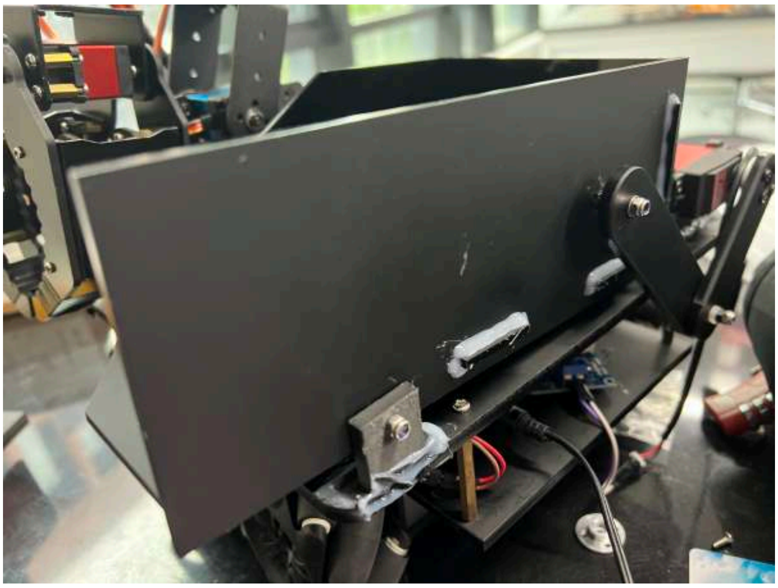












E Other related works

E.1 PID motor control (One motor example)

(or)

```

#include <FlexiTimer2.h>

#define Right_motor_go_2   46         // AIN1
#define Right_motor_back_2   47       // AIN2
#define Right_motor_pwm_2   9



/

    int.0     int.1    int.2    int.3    int.4    int.5
      2         3        21       20       29       18
  /
#define ENCODER_R2_A  21
```

```
#define ENCODER_R2_B 52

int value_R2;
String Target_Value;
int Velocity_R2,Count_R2=0;


float Velocity_KP =0.68, Velocity_KI =0, Velocity_KD = 7.2,Target=0;
int startPWM=0;
int PWM_Restrict=255;
void setup()
{
  Serial.begin(9600);
  Serial.println("/**********/");
  pinMode(ENCODER_R2_A,INPUT);
  pinMode(ENCODER_R2_B,INPUT);

  pinMode(Right_motor_go_2,OUTPUT);
  pinMode(Right_motor_back_2,OUTPUT);
  pinMode(Right_motor_pwm_2,OUTPUT);


  FlexiTimer2::set(5, control);
  FlexiTimer2::start ();     //
  attachInterrupt(2, READ_ENCODER_A_R2, CHANGE);
}

void loop()
{
 while(Serial.available()>0)          //
  {
    Target_Value=Serial.readString();  //
    Target=Target_Value.toFloat();     //,
    Serial.print(":");      //
    Serial.println(Target);
  }
  Serial.print(":");
  Serial.println(Velocity_R2);
}
void control()
{

        Velocity_R2=Count_R2;    //
        Count_R2=0;              //
        value_R2=Incremental_PI_A_R2(Velocity_R2,Target);
        Set_PWM_R2(value_R2);


}
void READ_ENCODER_A_R2()
{
    if (digitalRead(ENCODER_R2_A) == HIGH)
    {
     if (digitalRead(ENCODER_R2_B) == LOW)
       Count_R2++;  //
     else
       Count_R2--;
```

```
    }
    else
    {
     if (digitalRead(ENCODER_R2_B) == LOW)
     Count_R2--; //
     else
     Count_R2++;
    }

}
int Incremental_PI_A_R2 (int Encoder_R2,int Target)
{   float Bias_R2;
    float Bia_I_R2;
    static float PWM_R2=0,Last_bias_R2=0;
    Bias_R2=Target-Encoder_R2;                              //
    Bia_I_R2 += Bias_R2;
    PWM_R2+=Velocity_KP*Bias_R2 + Velocity_KI*Bia_I_R2 + Velocity_KD*(
Bias_R2-Last_bias_R2);   //PI
    if(PWM_R2>PWM_Restrict)PWM_R2=PWM_Restrict;
            //
    if(PWM_R2<-PWM_Restrict)PWM_R2=-PWM_Restrict;
            //
    Last_bias_R2=Bias_R2;                                   //
    return PWM_R2;                                       //
}


void Set_PWM_R2(int motora_R2)
{
  if (motora_R2 > 0)
  {
    digitalWrite(Right_motor_go_2,LOW);
    digitalWrite(Right_motor_back_2,HIGH);
    analogWrite(Right_motor_pwm_2, motora_R2+startPWM);
  }
  else if(motora_R2 == 0)
  {
    digitalWrite(Right_motor_go_2,LOW);
    digitalWrite(Right_motor_back_2,LOW);
  }
  else if (motora_R2 < 0)
  {
    digitalWrite(Right_motor_go_2,HIGH);
    digitalWrite(Right_motor_back_2,LOW);
    analogWrite(Right_motor_pwm_2, -motora_R2+startPWM);
  }
}
```

xleftmargin

E.2 PID tracking algorithm

(or)

```

#include "line.h"
#include "uart.h"
#include "motor.h"

```

```
    int lastError = 0;
boolean onoff = 0;
int val, cnt = 0, v[3];

const int maxspeed_high = 90;
const int minspeed_high = −90;
const int basespeed_high = 70;

const int maxspeed_low = 70;
const int basespeed_low = 50;
const int minspeed_low = −70;

float Kp = 0;
float Ki = 0;
float Kd = 0;
uint8_t multiP = 1;
uint8_t multiI = 1;
uint8_t multiD = 1;
uint8_t Kpfinal;
uint8_t Kifinal;
uint8_t Kdfinal;
int P;
int I;
int D;
float Pvalue;
float Ivalue;
float Dvalue;

/
   @brief

   @return int
  /
int ER_val()
{
  unsigned int temp_data[2] = { 0 };       //
  int error = 0;          //

  Read_Data(temp_data);

  if(((temp_data[0] >> 1)%2) == 1)    //
  {
    if(((temp_data[0] >> 3)%2) == 0)    //
    {
      if(temp_data[0]%2 == 0)
      {
        error = −temp_data[1];
      }
      else if(temp_data[0]%2 == 1)
      {
        error = temp_data[1];
      }

    }
  }
  //Serial.println(error);
  return error;
}
```

```
/ This void delimits each instruction.
The Arduino knows that for each instruction it will receive 2 bytes.

 /
void BT_set()
{
     BT_SERIAL.begin(9600);
}

void valuesread() {
  val = BT_SERIAL.read();
  cnt++;
  v[cnt] = val;
  if (cnt == 2)
    cnt = 0;
}

/ In this void the the 2 read values are assigned. /
void processing() {
  int a = v[1];
  if (a == 1) {
    Kp = v[2];
  }
  if (a == 2) {
    multiP = v[2];
  }
  if (a == 3) {
    Ki = v[2];
  }
  if (a == 4) {
    multiI = v[2];
  }
  if (a == 5) {
    Kd = v[2];
  }
  if (a == 6) {
    multiD = v[2];
  }
  if (a == 7) {
    onoff = v[2];
  }
  Serial.print("Kp:");
  Serial.println(Kp);
}

void robot_control_fast() {
  //0~7000
  //3500 ~ -3500
  //0,
  int error = ER_val();
  PID_fast(error);
}

/
   @brief PID

   @param error
```

```
 /
void PID_fast(int error) {
  int P = error;
  int I = I + error;
  int D = error - lastError;
  lastError = error;
//  Pvalue = (Kp/pow(10,multiP)) P;
//  Ivalue = (Ki/pow(10,multiI)) I;
//  Dvalue = (Kd/pow(10,multiD)) D;
  Pvalue = 0.41*P;
  Ivalue = 0.0041*I;
  Dvalue = 2*D;
  //errorerror
  float motorspeed = Pvalue + Ivalue + Dvalue;
  //pidpid
  int motorspeedL = basespeed_high + motorspeed;
  int motorspeedR = basespeed_high - motorspeed;
  //-100 ~ 150
  motorspeedL = constrain(motorspeedL, minspeed_high, maxspeed_high);
  motorspeedR = constrain(motorspeedR, minspeed_high, maxspeed_high);

  //Serial.print(motorspeedL); Serial.print(" "); Serial.println(
    motorspeedR);
  //
  speedcontrol(motorspeedL, motorspeedR);
}

void robot_control_slow() {
  //0~7000
  //3500 ~ -3500
  //0,
  int error = ER_val();
  PID_slow(error);
}

/
   @brief PID

   @param error
 /
void PID_slow(int error) {
  int P = error;
  int I = I + error;
  int D = error - lastError;
  lastError = error;
//  Pvalue = (Kp/pow(10,multiP)) P;
//  Ivalue = (Ki/pow(10,multiI)) I;
//  Dvalue = (Kd/pow(10,multiD)) D;
  Pvalue = 0.05*P;
  Ivalue = 0.005*I;
  Dvalue = 0.25*D;
  //errorerror
  float motorspeed = Pvalue + Ivalue + Dvalue;
  //pidpid
  int motorspeedL = basespeed_low + motorspeed;
  int motorspeedR = basespeed_low - motorspeed;
  //-100 ~ 150
  motorspeedL = constrain(motorspeedL, minspeed_low, maxspeed_low);
```

```
  motorspeedR = constrain(motorspeedR, minspeed_low, maxspeed_low);

  //Serial.print(motorspeedL); Serial.print(" "); Serial.println(motorspeedR);
  //
  speedcontrol(motorspeedL, motorspeedR);
}

void speedcontrol(int motL, int motR) {
  if (motL >= 0 && motR >= 0) {
    run_pid(motL, motR);
  }
  //0
  if (motL < 0 && motR >= 0) {
    //dreapta
    //
    motL = 0 - motL;
    spin_left_pid(motL, motR);

  }
  //0
  if (motL >= 0 && motR < 0) {
    //stanga
    //
    motR = 0 - motR;
    spin_right_pid(motL, motR);
  }
}

/
    @brief PID
  /
void BT_adj()
{
  if (BT_SERIAL.available()) {
    while(BT_SERIAL.available() == 0);
    valuesread();
    processing();
  }
  if(onoff == 1) {
    robot_control_fast();
//      robot_control_slow();
    //Serial.print("stop");
  }
  if(onoff == 0) {
    brake_encoder(1);
  }
}
```

xleftmargin