Team 6 - Handy hand

Shangyan Zhang, Shuohui Gao, Diwen Zhu

I. Background and motivation

Our project aims to build a mobile robotic arm controlled by hand gestures, in order to control the movement of the mobile car and to pick up and move objects to a given location. The idea of our project is that when we want to get something far away, we can just sit in front of the computer and control the robotic arm to help us get the items. Or when there is lots of trash that needs to be cleaned, we can control the robotic arm to throw them into the trash can. Despite the home housework purposes, we can also use it to deal with unreachable items under the sea or near dangerous places for further exploration.

There are two main parts of our project: gesture recognition and control of the robotic arm. These two parts have some previous examples online, for gesture recognition: https://www.youtube.com/watch?v=EgjwKM3KzGU, and for robotic arm: https://www.youtube.com/watch?v=Q3pCpVepCxk&t=218s and https://www.youtube.com/watch?v=LBNRGBY5zN8. Historical projects only involved one of these functions. For example, the robotic arm is usually controlled by given procedures or control handles. While our project is going to combine these two functions and make it more convenient for usage.

II. High level description

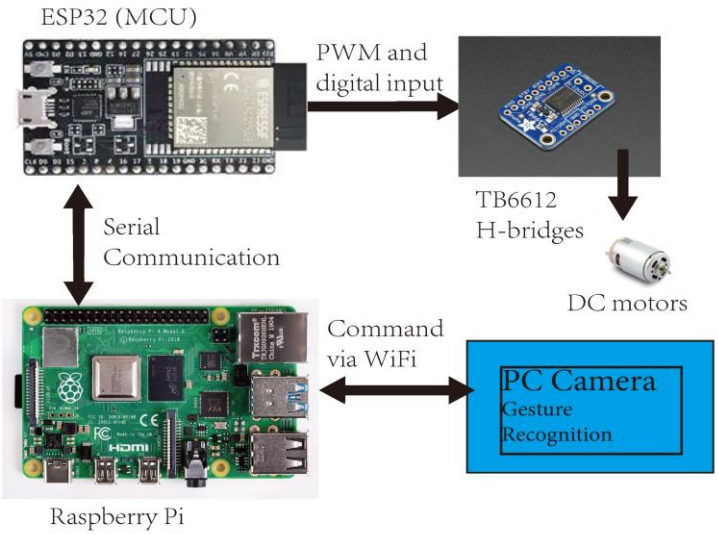


Figure 1. System architecture

The idea is to design a mobile robotic arm based on gesture recognition and computer vision to pick up and move objects to a given location. The hardware system uses a cart as a mounting platform, on which a robotic arm with a high degree of freedom is set up. The front of the car has a fixed camera to observe the environment, detect objects and calculate the distance

between the car and the object. The basic mode of use is that the user can use the designed gestures to move the car, control the robot arm, activate the distance-calculation function, and pick up the object to the specified position.

Figure 1 shows the system architecture diagram of this design. The system is divided into three main parts: PC, Raspberry Pi, and ESP32(MCU). The logic of the system is to recognize the user's gesture through PC and send it to the ESP32 through Raspberry Pi, and use the PWM signals to control six motors (two for the arm and four for the car) through the TB6612 H-bridges to achieve the function.

The functions of the PC are as follows:

1. Capture the user's gestures through the computer's camera, recognize these gestures with Mediapipe and transform them to digital result.

2. Send the recognition result to the Pi. Both hands have a total of 13 recognizable gestures, the function of the right hand is mainly to control the three different forward directions of the car and the control of robot arm, and the function of the left hand is mainly to control the three different backward directions of the car and change the speed mode of the car. Both hands can activate the distance-calculation function. Specific gesture definitions can be found in the Hand_definition.pdf file in the design folder.

The functions of the Raspberry Pi are as follows:

1. Receive the identification result from the PC.

2. Send the recognition result to the MCU.

3. Display the front environment in real-time.

4. When the distance-calculation function is activated, the results are displayed in the image.

The functions of the MCU are as follows:

According to the command delivered from Pi, run the Switch-case mode to perform the corresponding action.

There are two speed modes: fast mode and slow mode. The following figure shows the difference between the motion conditions of the robot arm and the car in different modes.

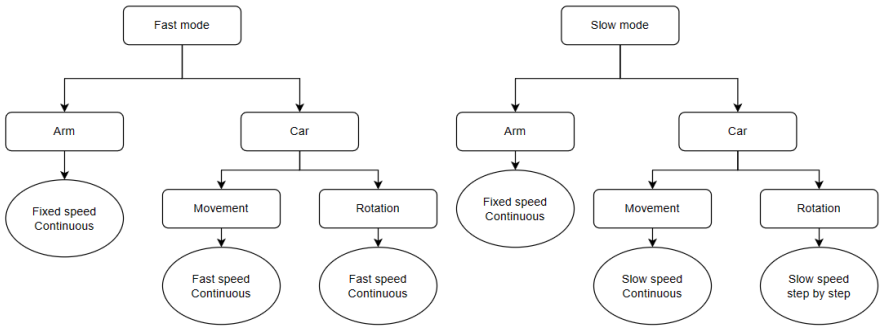


Figure 2. Fast & Slow mode

For each speed mode, we design the speed to be fixed, and the action of the robot arm (opening and closing of the pliers, rising and falling) is continuous. For the car, the movement is further divided into forward-backward and rotation. In fast mode, the car's forward-backward movement and rotation are rapid and continuous. In the slow mode, the forward-backward movement of the car is of low speed and continuous, while the rotation of the car is low speed and discontinuous.

The original intention of this design is to effectively complete the task in both normal movement and need to pick up items. The speed of the car in normal movement such as moving to a destination can improve efficiency. When the car needs to pick up items, we need to slowly adjust the car close to the item, which has higher requirements of precision for distance and angle, so we designed the car to move slowly in slow mode, and the steering can also be perfectly adjusted to the appropriate angle.

We also designed a module to measure distance, which aims to assist users in picking up items. The basic function is to measure the distance between the item and the camera, determine whether the item is directly in front of the car, and display the result in the real-time image to better assist the user in picking up the item.

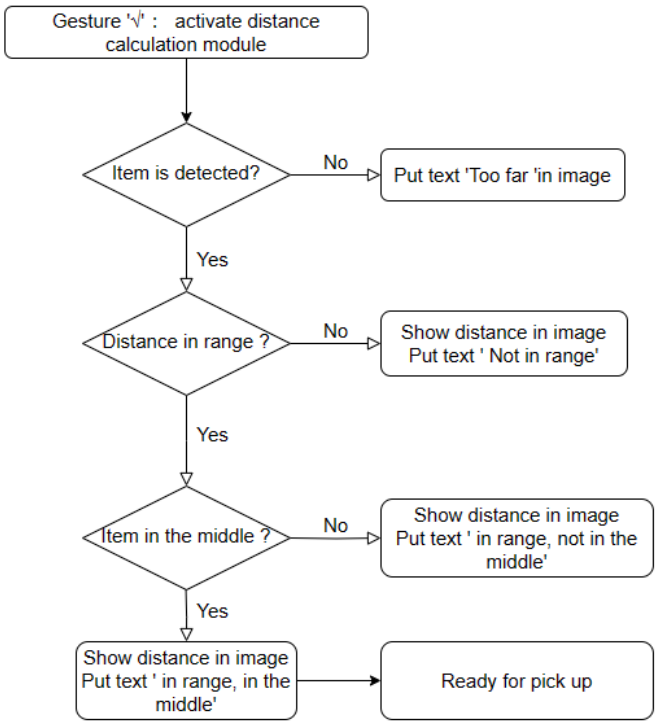


Figure 3. Different situations in the distance-calculation module

Figure 3 shows the corresponding image display in different situations when the distance-calculation module is activated. There are three conditions.

1. Determine whether the item is detected.

2. Once it is detected, determine whether the item is within the pickable range.

3. If the item is within the range, determine whether the item is in the middle.

When the object meets the above three conditions, it means that the cart is in the right place and ready to pick up, so the user can lower the robotic arm to pick up the item. Otherwise, the cart needs to be further adjusted until all conditions are met.

III. Important technical issues

Next, the three major parts of the system are analyzed in order, and the details that need to be paid special attention to when reproducing the work.

3.1 For PC side:

The main function of the PC is to recognize the user's gestures and transmit the recognition results to the Pi through WiFi.

We used Mediapipe and OpenCV libraries for gesture recognition: OpenCV was used to obtain real-time images for image processing, and Mediapipe was used to obtain 21 key nodes of hand joints. Users should be aware that we only recognize one hand at a time, in order to achieve a simple and easy-to-understand control scheme, so users should take care to hide the other hand when using it.

The specific process is as follows: First obtain the image, using Mediapipe to obtain the coordinates of the key nodes, the construction will draw the convex hull (the palm), while looking for the number of points located outside the convex hull (the highest point of the five fingers). Next, according to the function of Mediapipe, it determines whether the user uses the left hand or the right hand, and then first determines the number of points outside the convex hull and then judges the corresponding specific points to identify the correct gesture.

In order to achieve faster transmission speed, we discard the list and use the buffer to store the command and use the mode in the buffer as the final recognition result. Users can choose the balance according to their situation, if they want higher accuracy, the length of the buffer can be longer, if they want to update instructions faster, the buffer length can be shorter.

Because the processing speed of each frame is about 0.035s, we use every three frames to send instructions to the Pi end, if the user wants to speed up the transmission speed, you can use each frame to send instructions, but need to consider the movement time of the hardware, otherwise the hardware has not completed the corresponding task of the previous instruction received the next instruction.

PC and Raspberry Pi use sockets to establish WiFi wireless communication. The PC and Pi need to be on the same LAN, like home WiFi and mobile phone hotspots, because there is a firewall that prevents us from using the school WiFi. The PC needs to know the IP address of the Raspberry Pi and use the same port as the script on the Pi.

3.2 For Raspberry Pi side:

The Raspberry Pi receives the identification results from the PC over WiFi and transmits the results to the ESP32 using serial cable communication. It should be noted that the serial port of Pi needs to be selected according to the actual situation, and the baud rate should also be adjusted in time according to the speed of signal transmission.

The second is the parameter setting of the camera, because we are using the laboratory camera, the resolution is (640,480), if you use other cameras, remember to adjust the resolution. At the same time, we placed the camera upside down and made a 180-degree turn in the code so that the camera could be embedded in the front of the car and the cable would not obstruct the movement of the car. If you place the camera in the original direction, you need to remove the 180-degree turn code.

We followed the code from the laboratory work, converted the image from BGR to HSV, filtered it according to the H value of the picked item, and created a binary image. Finally, we calculated the center and side length of the object according to the binary image and used OpenCV to draw the edge. Since the H value of the item will change for the change of the environment, especially the lighting conditions, in a new testing environment, the user must first use the library of Matplotlib to observe the H value of the item in the image under this environment, and make sure that the color of the background and other environments should not be too close to the color of the item, otherwise, the error will be large. At the same time, the shape of the item should be regular and known, which is square in this experiment. If the user needs to pick up the sphere item or cuboid item, they need to change the code for calculating the side length correspondingly.

Finally, for the distance measurement module, since we only use one camera, we can use the principle of similar triangles to calculate the distance from the object to the camera. Because the focal length of the camera (F) is fixed, the focal length of the camera can also be calculated by experiment if different cameras are used. At the same time, we assume that the true width of the object (W) is also known, and by calculating the pixel width of the object in the image (P) through the steps above, we can calculate the approximate distance (D) from the equation of similar triangles D = F * W / P.

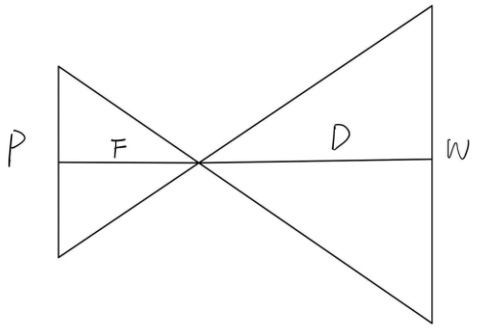


Figure 4. Sketch of the principle of distance measurement module

3.3 For ESP32:

We used functions from the TB6612FNG Arduino Library by SparkFun, which provides us with an easier way to control several motors at the same time with different PWM speed controls. We used #define to define the digital and PWM output pins of ESP32 that control three TB6612 chips. We have 6 motors in total; each TB6612 chip controls 2 motors. We modified the original Arduino Library to get rid of unnecessary pins that do not influence the wheel control. It is worth noting that when the ground environment is uneven or there is too much friction, the car may not move properly in slow mode, because low PWM leads to insufficient power dissipation. In this

case, the user needs to increase the speed threshold (PWM) in the Arduino program for the car to overcome rugged terrain in slow mode.

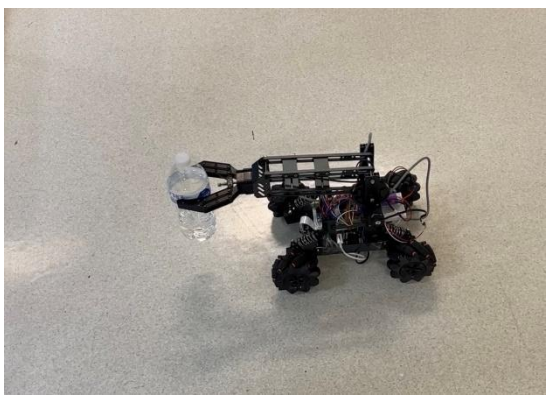
IV. System test and result analysis

From the very beginning to the present, we have gone through multiple iterations of the project. Firstly, we didn't assemble the entire car to do the test, but used the motors individually. We sent PWM signals to one motor and tested its performance. After successfully building the connection through the PC to the motors, we assembled the whole car and tested the whole car together.

At first, we set the motion of the motor discontinuous, which means that the car receives one command, then executes one motion and stops, and the interval between instructions is 1 second. This results in the movement of the car not being real-time.

We hope the movement of the car is continuous, so we changed the interval between instructions from 1 second to 0.1 second. Also, we modified the logic of the command, which let the car continuously conduct the last command until the next command came.

Then we tested the function of picking up items. We used a plastic bottle as the item to test its performance at first, since it's not heavy and has a larger coefficient of friction. After successfully picking the bottle up, we changed the item to a colored block, with a length of 5 cm, so that we could add the function of testing the distance between the camera on the car and the item. The situation of the robotic arm is shown in Figure 5.



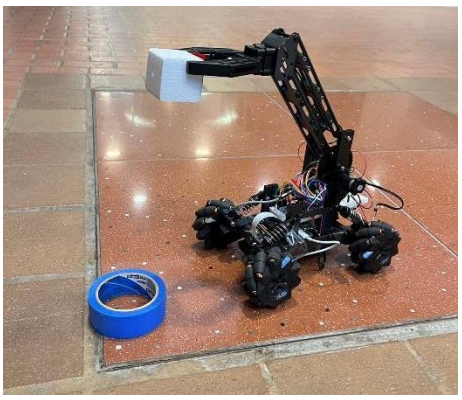


Figure 5. Actual tests of the robotic arm picking up items

If the car always moves continuously at a high speed, we cannot preciously control the car to pick up items at designated locations. Therefore, we added a function of slightly modifying the direction of the car, making it turn to the right direction of the item. Corresponding to 'fast-mode', we add 'slow-mode' to the car. In the slow mode, the car performs discontinuous steering, specifically moving for 0.2 seconds and stopping for 0.1 seconds. The logic of the speed mode is also described in Section 2. Here we attach a video (picking_up_bottle.mov) of the car picking up the bottle after fine-tuning in the zip file.

We also added the function of testing distance, so that we could tell the users what is the right time to pick up the item. We use a specific item -- a colored block, so that we can identify the location of an item by detecting its H value in HSV format, without being affected by the

environment. Since we should know the exact length of the item, we chose the foam block of a fixed side length as our item, and painted it green. Since the detection of the block is affected by the light and background, we tested its performance in the EECS building, adjusting its parameters to achieve the best detection effect. On the day of the final expo, we found a whiteboard behind the blocks, and adjusted the parameters again in the BBB building, to overcome the change of light condition. Also, since our robotic arm had a limitation in the height of its going down, we built a platform for our target blocks. The testing-distance function and the item on the expo day are shown below in Figures 6 and 7. In Figure 6 from left to right are the parameter tuning module, distance measurement with the camera on-board, and the user giving hand gestures of distance measurement.

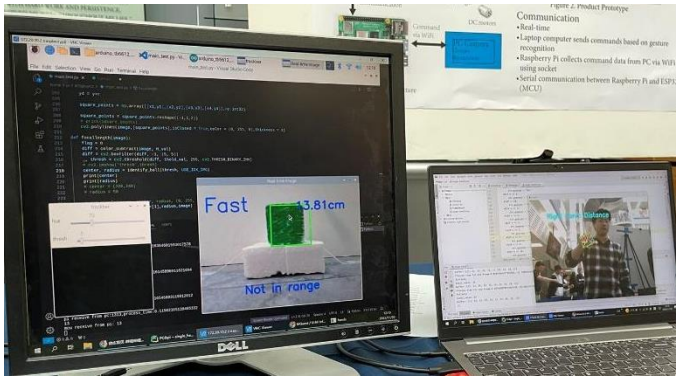


Figure 6. The result of the testing-distance function

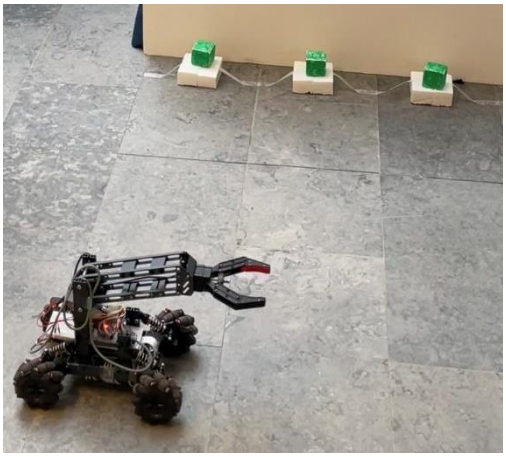


Figure 7. Green blocks and their platforms

V. Possible extensions and future works

For the H-bridge we used in our project, we used a breadboard to apply the TB6612 chips on it, and used wires to connect the H-bridge between ESP32 and the motors. The actual frame of the connection is shown below in Figure 8.

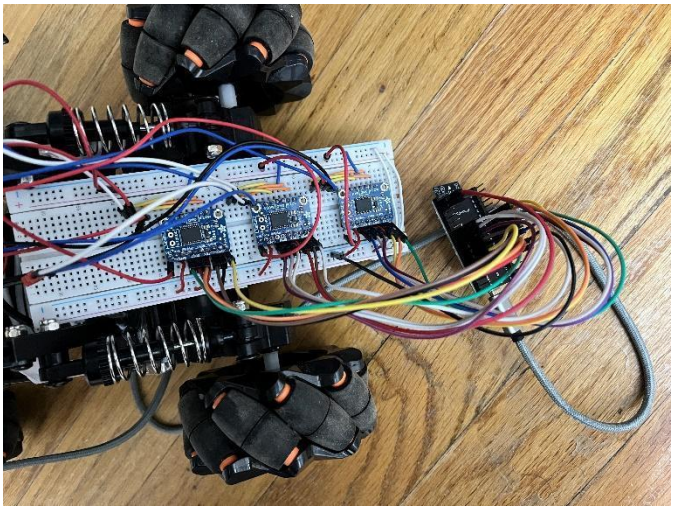


Figure 8. The connection between ESP32, H-bridge and motors

There are too many wires on our breadboard, which is messy. Therefore, we tried to design a PCB to make the whole circuit cleaner. We have finished the PCB design, and have received PCB from the manufacturer. Our possible future work is to finish soldering the PCB and test whether it can replace the breadboard that we are currently using.

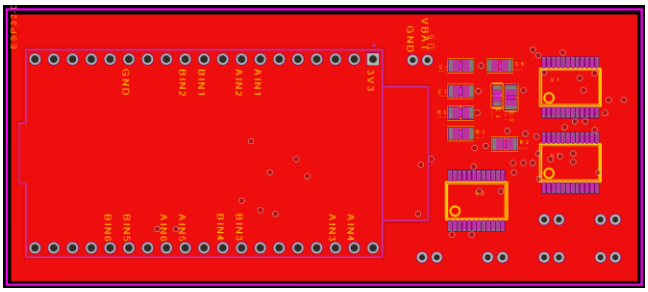


Figure 9. DXF file of our PCB

Using an ESP32 development board is an overkill for receiving serial commands and sending them to H-bridges. Other cheaper and simpler MCUs can be a better choice. Replacing the H-bridge-controlled robotic arm with encoders can allow us to get feedback on the position of the robotic arm and claw, which might give the robotic arm more possible features like automatic grab.

For the hand gestures, we can possibly do more work to make the gestures that control the movement and rotation of the car more intuitive. A possible extension to our code is to implement an object detection model like YOLO to identify the object that we are going to let the robotic arm grab, and even determine whether the robotic arm can grab the object or not. Another possible improvement that we can implement is to use the camera on the car and computer vision method to let the robotic arm adjust its position relative to the object automatically when we want the robotic arm to grab and the robotic arm is near the object.