OLLSCOIL TEICNEOLAÍOCHTA
BHAILE ÁTHA CLIATH
DUBLIN
TECHNOLOGICAL
UNIVERSITY DUBLIN

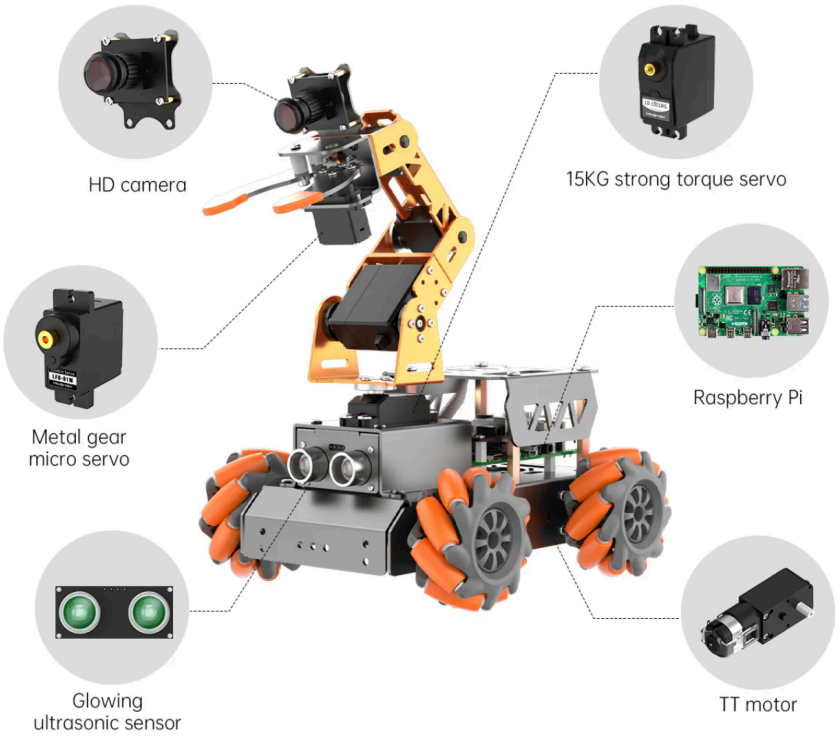
MasterPi Hiwonder

-

AI Vision Robot Arm with Mecanum Wheels Car Powered by Raspberry Pi

I - Introduction

MasterPi Intelligent robot is developed on Raspberry Pi. Equipped with mecanum chassis, arm robot and high definition camera, it is able to implement color sorting, object tracking, line follow, Intelligent transport. Combining with RGB glowing ultrasonic sensor, MasterPi can control light color and perform automatic obstacle avoidance.



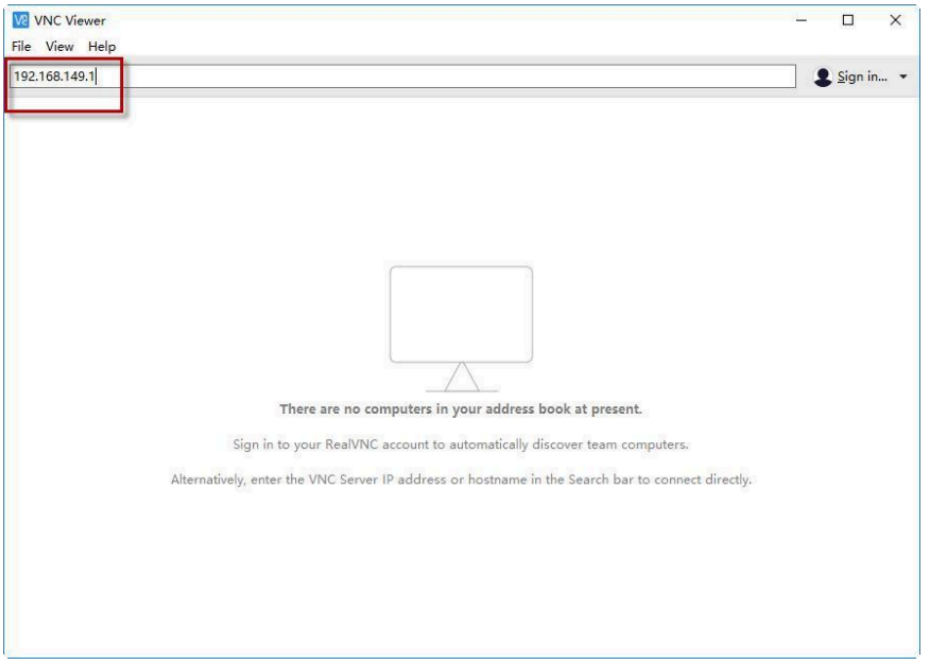
In addition, various sensors can be installed on MasterPi for more perception functions. We can use and set up all these features in Python3 with SDK availabled by Hiwonder.

a - Connect to the robot

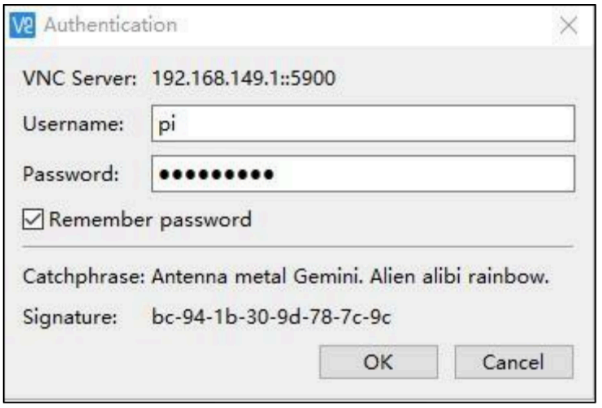
You can power the robot by putting *“ON”* the battery on the bottom of the robot. TheLED2 of Raspberry Pi expansion board will be on firstly and then the LED2will flash every 2 seconds, which means the robot is turned on successfully.
After turning on, Raspberry Pi will launch a Wi-Fi hotspot with a network named with the first letters “HW”. Click to connect.

Then you can install Real VNC viewer. VNC is a graphics remote control software. With VNC, we can control the Raspberry Pi directly from your computer through the hotspot created by Raspberry Pi. The VNC setup file is available at the at the root of the folder.

Next, you will learn how to use VNC. Enter the default IP address of the Raspberry Pi 192.168.149.1 in the VNC Viewer, and then press "Enter". If the software warns that the connection is not safe just click “Continue”.



Enter the password "raspberry" in the pop-up prompt box (if it requires you to enter an account name, use "pi"), then check the"Remember password" box, and then click "OK".



The Raspberry Desktop will now display on your computer's monitor.

b - Power supply of the robot

You need to use the batteries on the bottom of the robot. You can put on the Raspberry using port-c with a cable c, but that does not give enough energy to enable the Raspberry to support the Python program with the camera or the distance sensor. So obviously you need to use the battery available on the bottom of the Hiwonder robot.

Charger+
micro USB cable

Then to charge the battery you need to have a charger of it.

II - Development

For each next action with the robot, you can find her folder with the corresponding code and a video of her.

For the development in Python, you have two possibilities:

- You can develop on VNC using *Nano* or Geany.
- You can develop it on your local computer with the IDE of your choice. To transfer your program on the distant raspberry of the robot you could use *SSH*. To copy a program you can do the next command:
  To copy a file from your local computer to the raspberry:
  scp myProgram.py pi@192.168.149.1:/home/pi/MasterPi
  To copy a file from the Raspberry to your local machine:
  scp pi@192.168.149.1:/home/pi/MasterPi/program.py .

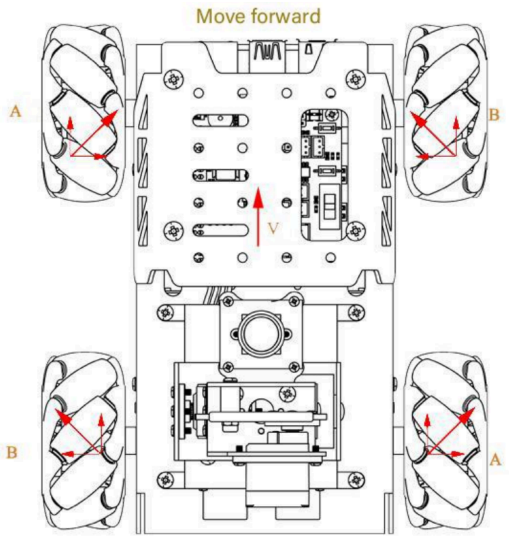
By the way, if you want to connect your local machine at the Hiwonder Robot you could connect yourself with the SSH protocol (with the same credential from VNC see here):
ssh pi@192.168.149.1

1 - Move Forward and stop

The path of this chapter: ./1 - Move forward and stop
You can copy this program in the raspberry:
scp 'move forward and stop.py' pi@192.168.149.1:/home/pi/MasterPi

Then you can execute this program with Python:
sudo python3 move\ forward\ and\ stop.py

Featuring 360°movement, flexibility, and stability, the mecanum wheel is a successful omnidirectional wheel. The combination of four mecanum wheels can be more flexible to realize the omnidirectional movement.



Any force can be decomposed into perpendicular vectors. Suppose the speed of wheel A and wheel B rotates at the same speed, a right force decomposed by wheel A and a left force decomposed by wheel B will counteract each other, and the direction of resultant velocity is forward.

```
import HiwonderSDK.mecanum as mecanum
chassis = mecanum.MecanumChassis()
```

The control module related to mecanum wheel chassis. The call to *mecanum.MecanumChassis()* constructor creates an object of the MecanumChassis class and assigns it to the chassis variable.

chassis.set_velocity(speed,direction,rotation speed)

- speed [-100; 100] (counterclockwise - clockwise)
- direction [0; 360]
    - 90° forward
    - 270° backward
    - 0° right
    - 180° left
- rotation speed [-2; 2] (counterclockwise - clockwise)

2 - Go forward and stop when you detect something

The path of this chapter: ./2 - Go forward and stop when you detect something

The ultrasonic sensor can range the distance of the front object and control the

movement of the car. If the distance of the front object exceeds the detected distance, the car will move forward; if the distance is less than the detected distance, the car will stop moving and turn to avoid the obstacle.

```
if __isRunning:
    # if the current distance is more small than the limit
    if distance <= Threshold:
        if wait:
            wait = False
            forward = True
            stopMotor = True
            chassis.set_velocity((5* speed/6),90,0)
            time.sleep(0.2)
```

It reads the distance from the ultrasonic sensor and processes it to filter out noise using a rolling mean and standard deviation. If the robot runs and an obstacle is detected within the threshold distance, it gradually reduces speed and stops. If no obstacle is detected, it moves forward at the set speed. We check whether the current distance meets the threshold set during initialization. Based on the result, different actions can be executed subsequently. In this context, the value "15.0" inside the *Threshold* parameter corresponds to the *threshold* of 15 cm.

3 - Go forward at a random starting speed

The path of this chapter: ./3 - Go forward at a random starting speed

```
chassis.set_velocity(random.randint( a: 30,  b: 100),90,0)
time.sleep(5)
chassis.set_velocity(0,0,0)
time.sleep(5)
```

With *random.randint()* at each iteration of the *while* boucle we apply a velocity between 30 and 100 on the robot.

4 - Go forward only if the surface is RED

The path of this chapter: ./4 - Go forward only if the surface is RED

We move forward only when a red surface is detected. It uses computer vision to process the images captured by a camera and identify the color of the surface.

```
if detect_color != 'None' and start_pick_up:

    set_rgb(detect_color)
    setBuzzer(0.1)

    if detect_color == 'red' :
        chassis.set_velocity(70,90,0)
        AK.setPitchRangeMoving((0, 6, 18), 0,-90, 90, 500)
        detect_color = 'None'
        start_pick_up = False
        set_rgb(detect_color)

    else:
        chassis.set_velocity(0,90,0)
        AK.setPitchRangeMoving((0, 6, 18), 0,-90, 90, 500)
        detect_color = 'None'
        start_pick_up = False
        set_rgb(detect_color)
```

If we detected a color and if the color detected is red we apply a vitesse to the chassis of the car.

5 - Go forward and go fast on RED and slow on BLACK

The path of this chapter: ./5 - Go forward and go fast on RED and slow on BLACK

```
if detect_color == 'red' :
    chassis.set_velocity(100,90,0)
    AK.setPitchRangeMoving((0, 6, 18), 0,-90, 90, 500)
    detect_color = 'None'
    start_pick_up = False
    set_rgb(detect_color)

if detect_color == 'black' :
    chassis.set_velocity(40,90,0)
    AK.setPitchRangeMoving((0, 6, 18), 0,-90, 90, 500)
    detect_color = 'None'
    start_pick_up = False
    set_rgb(detect_color)
```

If we detect a color and the color detected is red we apply a vitesse to the chassis of the car and if we detect a black color we apply less velocity and else we stop.

6 - Follow a line straight

The path of this chapter: ./6 - Follow a line straight

This program allows the robot to follow a colored line with the camera, detecting the line. The robot will move forward when the line is centered in the frame and stop if the line is not detected or deviates significantly from the center.

```
if line_centerx != -1:
    if abs(line_centerx - img_centerx) < 20:  # If the line is approximately in the center
        chassis.set_velocity(50,90,0)
    else:
        MotorStop()
else:
    MotorStop()
```

While we detect a line in the center of the camera, we go forward.

7 - Follow a line curved

The path of this chapter: ./7 - Follow a line curved

While we detect a line we follow the line, if the line isn't in the center of the camera we adjust the velocity of the wheel to center the car on the line. We make tree zone/rectangle to see and compute the trajectory of the line.

This includes binarizing the image to reduce interference andmake it smoother. Then obtain the contour with the largest area and theminimum enclosing circle of the target, and calculate the central coordinatesof the target. Lastly, PID algorithm is used to control the chassis accordingtothecentre coordinates.

The inRange() function from cv2 library is used to perform image **binarization processing**:

```
frame_mask = cv2.inRange(frame_lab,
                         (lab_data[i]['min'][0],
                          lab_data[i]['min'][1],
                          lab_data[i]['min'][2]),
                         (lab_data[i]['max'][0],
                          lab_data[i]['max'][1],
                          lab_data[i]['max'][2]))
```

The first parameter “frame_lab” is the input image.
The second parameter “tuple(color_range['min'])” is the lower limit of threshold.
The third parameter “tuple(color_range['max'])” is the upper limit of threshold.

To obtain the minimum bounding rectangle of the target contour using theminAreaRect() function from the cv2 library, and then get the coordinates of itsfour vertices using the boxPoints() function. Afterwards, you can calculatethecoordinates of the center point by using the vertex coordinates of the rectangle.

```
if cnt_large is not None:
    rect = cv2.minAreaRect(cnt_large)
    box = np.int0(cv2.boxPoints(rect))
    for i in range(4):
        box[i, 1] = box[i, 1] + (n - 1)*roi_h + roi[0][0]
        box[i, 1] = int(Misc.map(box[i, 1], 0, size[1], 0, img_h))
    for i in range(4):
        box[i, 0] = int(Misc.map(box[i, 0], 0, size[0], 0, img_w))

    cv2.drawContours(img, [box], -1, (0,0,255,255), 2)

    pt1_x, pt1_y = box[0, 0], box[0, 1]
    pt3_x, pt3_y = box[2, 0], box[2, 1]
    center_x, center_y = (pt1_x + pt3_x) / 2, (pt1_y + pt3_y) / 2
    cv2.circle(img, (int(center_x), int(center_y)), 5, (0,0,255), -1)
    center_.append([center_x, center_y])

    centroid_x_sum += center_x * r[4]
    weight_sum += r[4]
```

After the image processing are complete, the Board.setMotor() function is called to control the movement of the motor on robot.
The *Board.setMotor()* function is used to control motor. Take an exampleof thec ode “ *Board.setMotor(1, int(50-base_speed))*, the meaning of the parameters in parentheses is as follow: The first parameter “1” is the sequential number of the motor, representing motor 1. The second parameter *int(50-base_speed)* is the speed, which represents the current movement speed of the robot plus or minus the speed of the PID compensation.

8 - Go around an obstacle

The path of this chapter: ./8 - Go around an obstacle

If the distance of the front object exceeds the detected distance, the car will move forward; if the distance is less than the detected distance, the car will stop moving and turn to avoid the obstacle.

```
if line_centerx != -1:

    num = (line_centerx - img_centerx)
    if abs(num) <= 5:
        pitch_pid.SetPoint = num
    else:
        pitch_pid.SetPoint = 0
    pitch_pid.update(num)
    tmp = pitch_pid.output
    tmp = 100 if tmp > 100 else tmp
    tmp = -100 if tmp < -100 else tmp
    base_speed = Misc.map(tmp, -100, 100, -50, 50)
    Board.setMotor(1, int(50-base_speed))
    Board.setMotor(2, int(50+base_speed))
    Board.setMotor(3, int(50-base_speed))
    Board.setMotor(4, int(50+base_speed))

else:
    MotorStop()
    time.sleep(0.01)
```

After starting game, the distance of ranged object will be displayed on thetransmitted screen. When the distance between car and detected object islessthan 30cm, the car will stop moving; when the distance is more than 30cm, thecar will move forward.

```
if distance ⩽ Threshold:
    if turn:
        turn = False
        forward = True
        stopMotor = True
        chassis.set_velocity(0,90,-0.5)
        time.sleep(0.5)

else:
    if forward:
        turn = True
        forward = False
        stopMotor = True
        chassis.set_velocity(speed,90,0)
```

If the robot runs and an obstacle is detected within the threshold distance, we apply a rotation in counterclockwise to make a turn on the right of the car while the distance is under 30 cm.

9 - Control the arm, all motors

```
x_dis = 500 if x_dis < 500 else x_dis
x_dis = 2500 if x_dis > 2500 else x_dis

y_dis = 500 if y_dis < 500 else y_dis
y_dis = 2500 if y_dis > 2500 else y_dis

while True:
    Board.setPWMServosPulse([20, 2, 3,int(y_dis), 6,int(x_dis)])
    time.sleep(0.5)
```

This command adjusts the arm servomotors to follow the object in x and y directions.
The possible values for x_dis and y_dis in the robotic arm tracking program are defined in the code to ensure efficient and safe arm movement. The values of x_dis and y_dis are limited to avoid extreme movements that could damage the robotic arm.

```
AK = ArmIK()

if __name__ == '__main__':
    while True:
        AK.setPitchRangeMoving((0, 7, 12), -50, -90, 0, 1500)
        time.sleep(0.5)
```

position (0, 7, 12) :

- x = 0: Target position in millimeters along X axis.
- y = 7: Target position in millimeters along the Y axis.
- z = 12: Target position in millimeters along the Z axis.

alpha = -50 :

- Angle of rotation around X axis, in degrees. An angle of -50 degrees means that the arm is tilted downwards in relation to the X axis.

beta = -90 :

- Angle of rotation around the Y axis, in degrees. An angle of -90 degrees means that the arm is oriented downwards in relation to the Y axis.

gamma = 0 :

- Angle of rotation around the Z axis, in degrees. An angle of 0 degrees means there is no rotation around the Z axis.

time = 1500 :

- Time in milliseconds to reach target position. Here, the arm will take 1500 milliseconds (1.5 seconds) to move to the target position.

III - Conclusion

1 - Cost of the Robot

The robot we currently have uses a Raspberry 4 model b with 4 GB of RAM. So on the official website of Hiwonder, the price of a robot identical to the current robot is $289.99 so approximately €266,89.

Official website:
https://www.hiwonder.com/collections/raspberry-pi/products/masterpi?variant=40948133101655

Next the other price with other raspberry formats available:

- Raspberry 4 (4 GB): $289.99
- Raspberry 4 (8 GB): $329.99
- Raspberry 5 (4 GB): $349.99
- Raspberry 5 (8 GB): $379.99

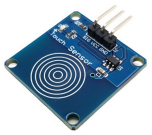
The robot is also available on Amazon with a Raspberry 4 or without Raspberry:
https://www.amazon.com/HIWONDER-MasterPi-Mecanum-Raspberry-Beginners/dp/B0CP97SZPJ

2 - Additional sensor

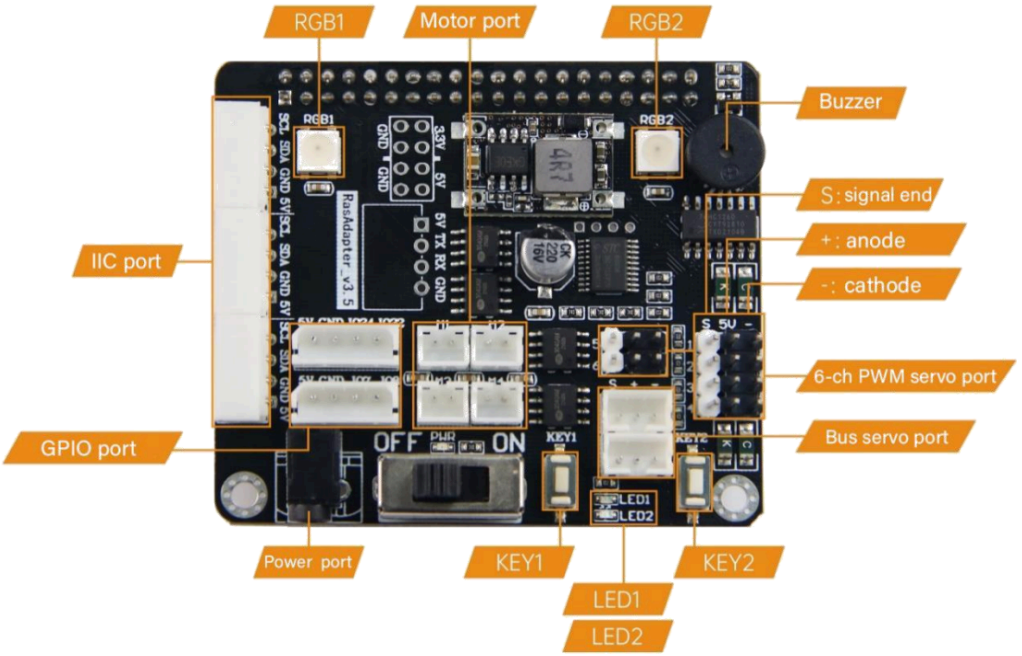
For the secondary devolvement on the robot, we need to use the expansion board.

We could use the slots of IIC port and GPIO port, only **4 ports** are free.

Touch Sensor:



Sensor:
https://www.digikey.ie/en/products/detail/sunfounder/ST0246/22116807?utm_adgroup=&utm_term=&productid=22116807&utm_content=&gad_source=1
Example of code: https://newbiely.com/tutorials/raspberry-pi/raspberry-pi-touch-sensor



Distance Sensor:

Sensor:
https://ie.rs-online.com/web/p/bbc-micro-bit-add-ons/2153181?matchtype=&&s_kwcid=AL!14853!3!!!x!!&gad_source=1

I already have an example of code for this sensor. We will need to precise the port of the capture and not use the SDK of Hiwonder:

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time
try:
    GPIO.setmode(GPIO.BOARD)
    PIN_TRIGGER = 7
    PIN_ECHO = 11
    GPIO.setup(PIN_TRIGGER, GPIO.OUT)
    GPIO.setup(PIN_ECHO, GPIO.IN)
    GPIO.output(PIN_TRIGGER, GPIO.LOW)
    print "Waiting for sensor to settle"
    time.sleep(2)
    print "Calculating distance"
    GPIO.output(PIN_TRIGGER, GPIO.HIGH)
    time.sleep(0.00001)
    GPIO.output(PIN_TRIGGER, GPIO.LOW)
    while GPIO.input(PIN_ECHO)==0:
        pulse_start_time = time.time()
    while GPIO.input(PIN_ECHO)==1:
        pulse_end_time = time.time()
    pulse_duration = pulse_end_time - pulse_start_time
    distance = round(pulse_duration * 17150, 2)
    print "Distance:",distance,"cm"
finally:
    GPIO.cleanup()
```

Sound sensor:

Sensor: https://ie.rs-online.com/web/p/sensor-development-tools/1743255?gb=s
Example of code: https://newbiely.com/tutorials/raspberry-pi/raspberry-pi-sound-sensor

Gyro sensor:

Gyro sensor:
https://www.digikey.ie/en/products/detail/dfrobot/SEN0409/14824962?utm_adgroup=&utm_term=&productid=14824962&utm_content=&gad_source=1
or

https://www.digikey.ie/en/products/detail/adafruit-industries-llc/1018/4990760?utm_adgroup=&utm_term=&productid=4990760&utm_content=&gad_source=1
Example of code:
https://docs.sunfounder.com/projects/umsk/en/latest/05_raspberry_pi/pi_lesson05_mpu6050.html

3 - Other feature

- **Object Recognition and Classification:** Implement advanced machine learning algorithms to recognize and classify objects in real time. This can enable the robot to perform more complex tasks such as sorting items based on type or quality. Maybe use a distant server that uses an IA, because the Raspberry can't manage the robot and a model of data at the same time.

- **Localization and Mapping:** Integrate algorithms to enable the robot to map its environment and navigate autonomously without pre-defined paths.

- **Dynamic Obstacle Avoidance:** Enhance the robot's ability to detect and avoid obstacles by adapting its movement to the obstacle's position (like turning right or left for example).

- **Use Pliers:** Apply fine motor skills, enabling the robot to perform delicate tasks such as picking up small components.

- **IoT Connectivity:** Enable the robot to connect with other smart devices and systems, allowing it to be part of an integrated IoT ecosystem for smart homes or industrial automation.

- **Visual Programming Interface:** Create a visual programming interface to make it easier for users, especially beginners and educators, to program and control the robot.

- **Adding sensor:** Add a sensor on the robot to better understand the universe around it, like detect when the robot will fall for examle.

- **Follow the line with arm and wheels:** Follow the line with wheels and the arm, to try to have a movement more fluid and more human. Follow a line with an arm and wheel could move quickly in a curved line.