* + 1. System Integration
       1. Patio 1

The patio 1 scene layout and partial action track is shown in Figure 1. The patio 1 task can be divided into three general phases. As for the first phase, the robot will start from the starting point (green) and follow the trail to the bridge (blue) by identifying the coloured tiles. As for the second phase, the robot detects the obstacles (yellow) to its right, makes a turn and then goes up and down the bridge. As for the third phase, after coming down from the bridge, the robot recognises the coloured tiles (blue) and follows the path to the gate (purple). Once the gate has been detected, the robot will stop automatically after a short period of time.

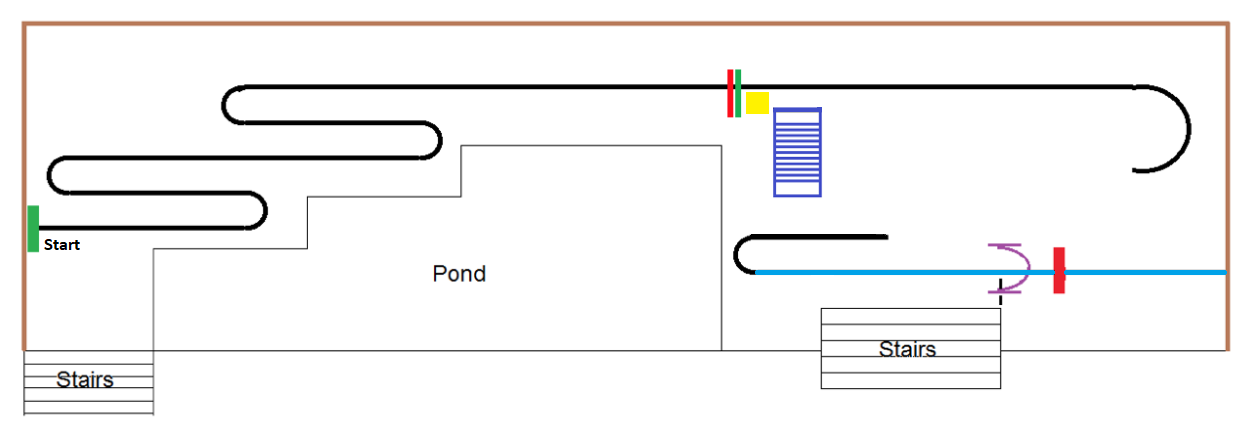


Figure . Patio 1 scene layout and partial action track.

The patio 1 flow diagram is shown in Figure 9, which corresponds to the control logic of the MCU code.

The first loop shown in the flowchart corresponds to the first stage of patio 1: OpenMV continuously identifies the coloured tiles and sends the processed information to the MCU, which in turn sends control signals to the motors for line tracking. During this loop, the two ultrasonic sensors on the right side of the robot are also constantly checking the distance to the obstacle. When the distance detected by both ultrasonic sensors is less than 20 cm at the same time, i.e. when the robot finishes the first phase of patrolling and comes to the yellow obstacle shown in Figure 1, the robot exits the first loop.

 地上的行李箱

低可信度描述已自动生成

Figure . Line tracking in phase 1 of patio 1.

When the robot detects the obstacle, the robot will again travel straight ahead for 0.7 seconds to a position roughly on the horizontal median of the bridge. Then the robot itself turns 90 degrees to the right in place using the angular accelerometer and gyroscope.

 桌子上摆放着黑色的机器

低可信度描述已自动生成

Figure . Ultrasonic recognition of obstacles and 90 degree turns.

The bridge is rectangular in shape and four vertical upward rods exist at the four corners of the rectangle. After a 90 degree turn, the direction of travel of the robot is not necessarily perpendicular to the plane formed by the two nearest poles to the upper bridge slope, and there may be a slight error. This error is magnified by the fact that the robot is positioned at a considerable distance from the upper bridge slope, and after a period of travel. If this error is not corrected, the robot may not be able to move up the upper bridge slope, or may travel in a bad direction of travel on the upper bridge slope.

Therefore, after the robot has turned 90 degrees, we let it travel forward again for 1.2s and stopped, adjusting the robot's forward direction in situ by means of the ultrasonic sensors on the right side (the plane of the obstacle is parallel to the plane of the right side of the bridge).



Figure . Traveling forward again for 1.2s and stopped, adjusting the forward direction.

After adjusting the forward direction, the robot enters the second loop shown in the flowchart. In this loop, the robot constantly adjusts itself via the JY60 (the angular accelerometer and gyroscope) so that it can move in a straight line. When either of the two ultrasonic sensors on the right side detects the rod on the bridge, the robot will jump out of this loop.

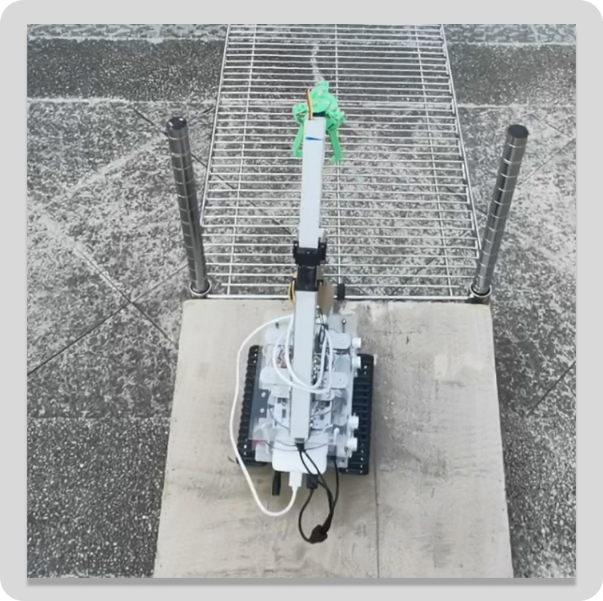


Figure . Getting on the bridge.

The bridge deck between the ramps will consist of barbed wire. Once the robot is on the bridge, the gaps between the tracks are partially embedded in the wire mesh, thus again correcting the robot's direction of travel. So, we can just keep the robot moving in a straight line over the bridge.



Figure . Crossing the bridge.

Once the robot is off the bridge and recognize the coloured tiles, it goes into the third loop shown in the flowchart. The third loop is similar to the first loop: OpenMV continuously identifies the coloured tiles and sends the processed information to the MCU, which in turn sends control signals to the motors for line tracking. During this loop, the two ultrasonic sensors on the right side of the robot are also constantly checking the distance to the gate frames. When the distance detected by one of the ultrasonic sensors is less than 30 cm, i.e. when the robot has crossed the gate (purple) shown in Figure 1, the robot exits the third loop.

 图片包含 建筑, 猫, 厨房, 街道

描述已自动生成

Figure . Line tracking after getting off the bridge.

Finally, after passing through the gate, the robot will keep travelling forward for 2s and stop.

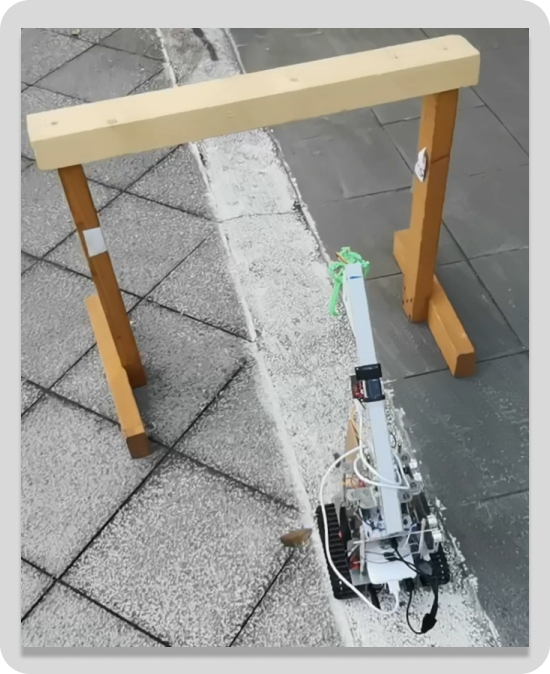


Figure . Passing through the gate.

图示

描述已自动生成

Figure . The patio 1 flow diagram.

* + - 1. Patio 2

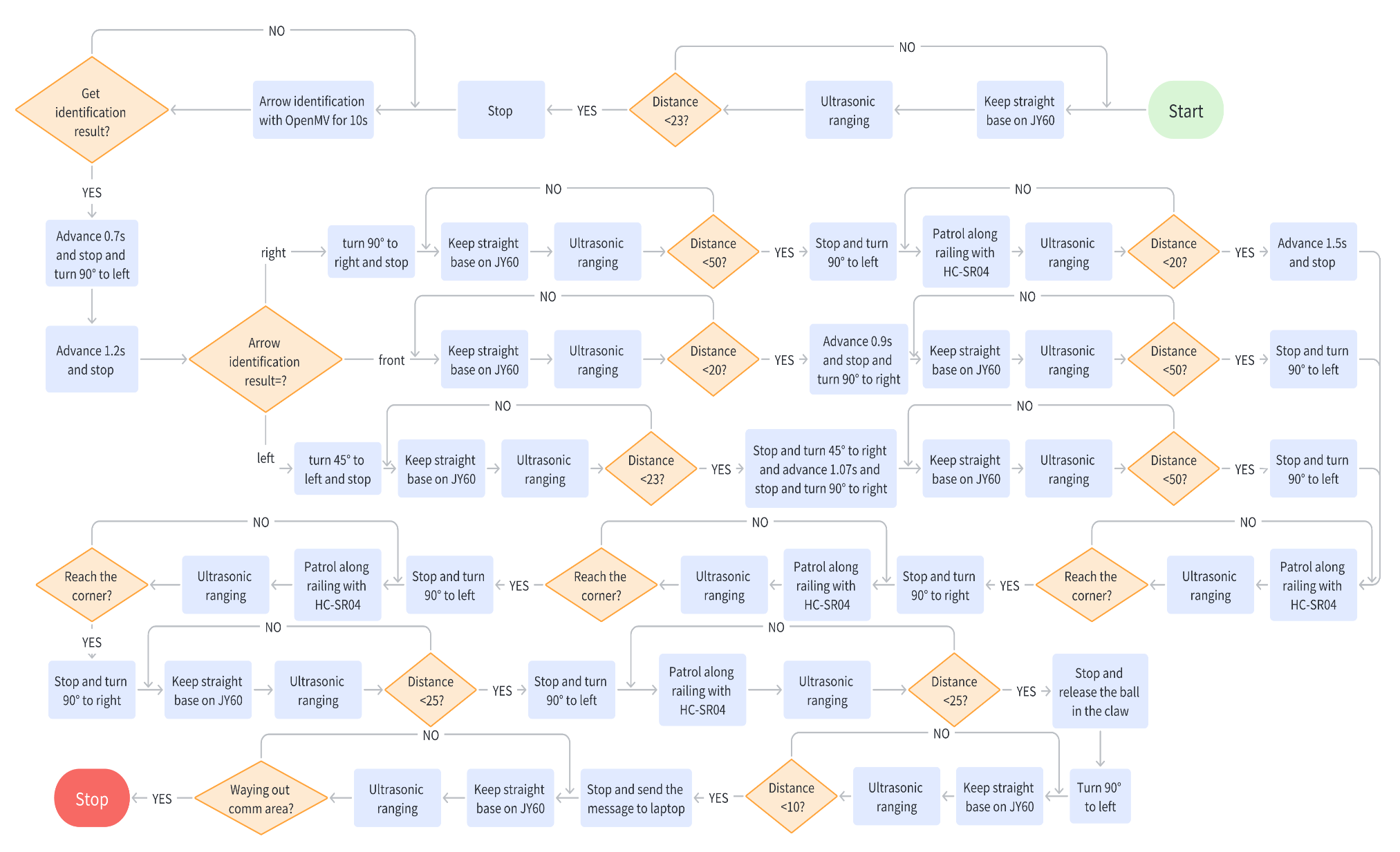
The patio 2 tracked route is shown in Figure 10. Starting from the starting point (green), the robot will move to the yellow square where it would scan and identify the shapes on the squares to determine the direction and then knock out a bracket with matching shape in positions 1, 2 and 3.

图示

描述已自动生成

Figure . Patio 2 tracked route.

Figure . The patio 2 flow diagram



* 1. Discussion
     1. Arrow Identification

In task1, the robot will stop in front of the sign and identify the arrow on the sign, but OpenMV may identify the arrow incorrectly. In order to ensure that OpenMV sends the correct arrow recognition result to the MCU, we extend the time that the robot stays in front of the sign. This allows OpenMV to perform multiple recognitions, of which the one with the highest number of occurrences is the final result. The robot then behaves accordingly to this result.

In addition, we have improved the recognition algorithm to increase the recognition accuracy. In the final solution, the previous grey-scale graphics interception algorithm was replaced as the binary algorithm for recognition. The algorithm binarizes the images captured by the camera, which significantly improves recognition accuracy. This is a common method of data enhancement. The Figure 12 shows the result of image binarization.

图示, 形状, 箭头

描述已自动生成

Figure Example for image binarization

* + 1. Line Patrol
       1. Straight Line Movement

In our field tests, we found that if the robot is only able to move in a straight line by controlling the rotation of the wheels, the robot will deviate from the straight direction after a certain period of time. To solve this problem, we chose to use the accelerometer to assist with this function: before starting to move in a straight line, the direction of the robot was read from the accelerometer as the initial direction. After this, the robot would adjust the wheels according to the difference between the real time direction and the initial direction to ensure that the robot could move in a straight line.

In Figure 13, this adjusting process was shown in detail. At the beginning, the robot is set to remain stationary for a short period to obtain the initial direction (denoted as ‘a’ in figure and code). While moving forward, robot will adjust the real-time orientation (defined as ‘yaw1’) to ‘a’. To achieve this, the robot adjusts the speed of its wheels based on the orientation deviation: increasing the speed of the wheel on the side of the deviation and decreasing the speed of the other wheel, both with the same amount of variation (defined as ‘Coeff’). For example, when the ‘yaw1’ is deviated to the right of the ‘a’, the robot increases the speed of the right wheel and decreases the speed of the left wheel. The ‘Coeff’ is calculated as follow:



Since the rotation speed of motor is control by the PWM signal, the speed was represented as the duty cycle of PWM. The setting of the two parameters of the last function is obtained by our group through a large number of field tests, which has good robustness in the field. At the same time, to ensure that the robot will not turn around in place because of the large speed difference between the tracks, the value of ‘Coeff’ should below 0.1.

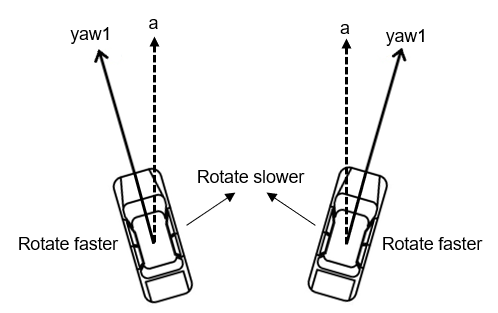


Figure . Adjusting the direction while moving forward

* + - 1. Patrolling along the railings

As shown in Figure 10, in some stages, the robot will patrol the line along the railings, this is achieved by means of ultrasonic sensors mounted on the right side of the trolley. In the original solution, the robot would stop and reorient itself to be parallel to the railing before continuing on its way. However, after field tests this was found to be very time consuming. In the end we chose to reorient the robot as it travelled and to ensure that there was a fixed distance between the robot and the railing. With this improvement, the robot was able to do a good job of moving along the railings.

The algorithm to implement this function is similar to the one mentioned in the previous section. The main difference is that the orientation deviation is replaced by the difference in distance from two ultrasonic sensors to the railing, when calculating the amount of speed variation ‘Coeff’. Furthermore, we expected that the robot could maintain a pre-determined distance from the railing as it patrols along it. The robot controls its wheel speeds based on the distance (measured by ultrasonic sensor) from the robot to the railing, slowing down the wheels on the side near the railing when the distance is greater than the set distance, and speeding them up when the distance is less. Conversely, the wheel speed on the opposite side is adjusted in the reverse manner: slowing down when the distance to the railing is less than the set distance and speeding up when the distance is greater.

* + - 1. Crossing the stone road

The robot needed to cross the stone road after releasing the tennis ball. The area of the stone road (blue) and the route of travel in this area (red arrow) is shown in Figure 14. When the robot is running on the stone road, large bumps occur. At this point the ultrasonic modules' recognition of the distance to side objects can be very unstable and the robot has difficulty using this data to make adjustments to its direction of travel. As the robot only needs to travel in a straight line on the gravel road (except for turns), we have chosen to use the accelerometer directly to assist the robot to travel in a straight line through this area. The specific algorithm has mentioned in 2.2.1.

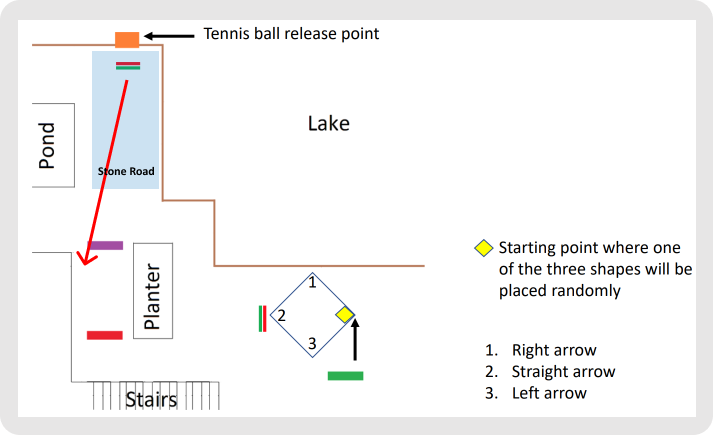


Figure . Patio 2 – the stone road.

* + - 1. Turning

At first, we fixed the parameters that control the robot's turns. However, we found that as the running time increased and the battery power decreased, the robot would lose power. In order to get it to turn in the right direction, these parameters need to be modified. This suggests that these parameters will be time dependent. To solve this problem, we chose to use the angle information from the accelerometer to assist the robot in turning.