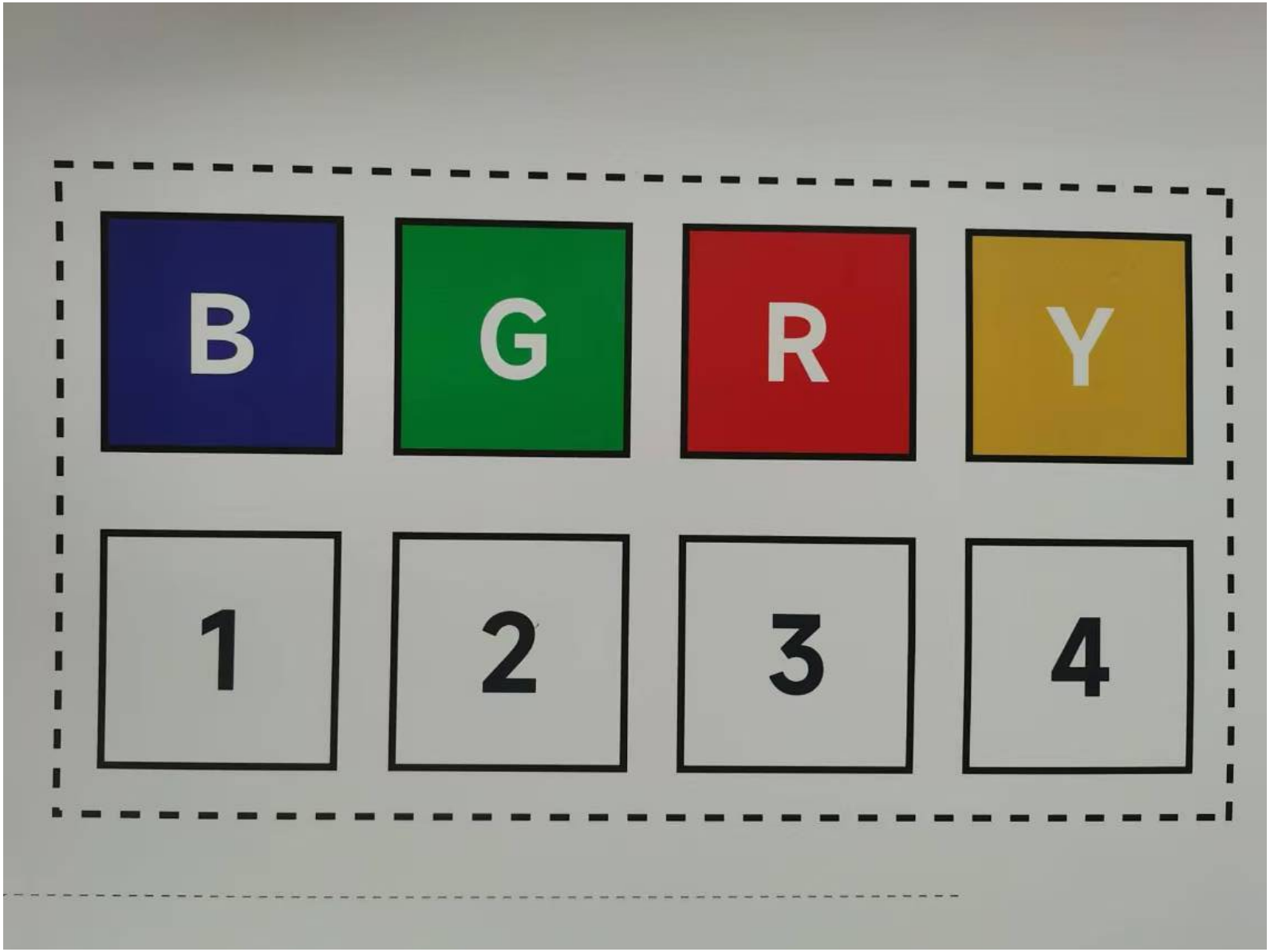
# Gesture recognition stacking blocks

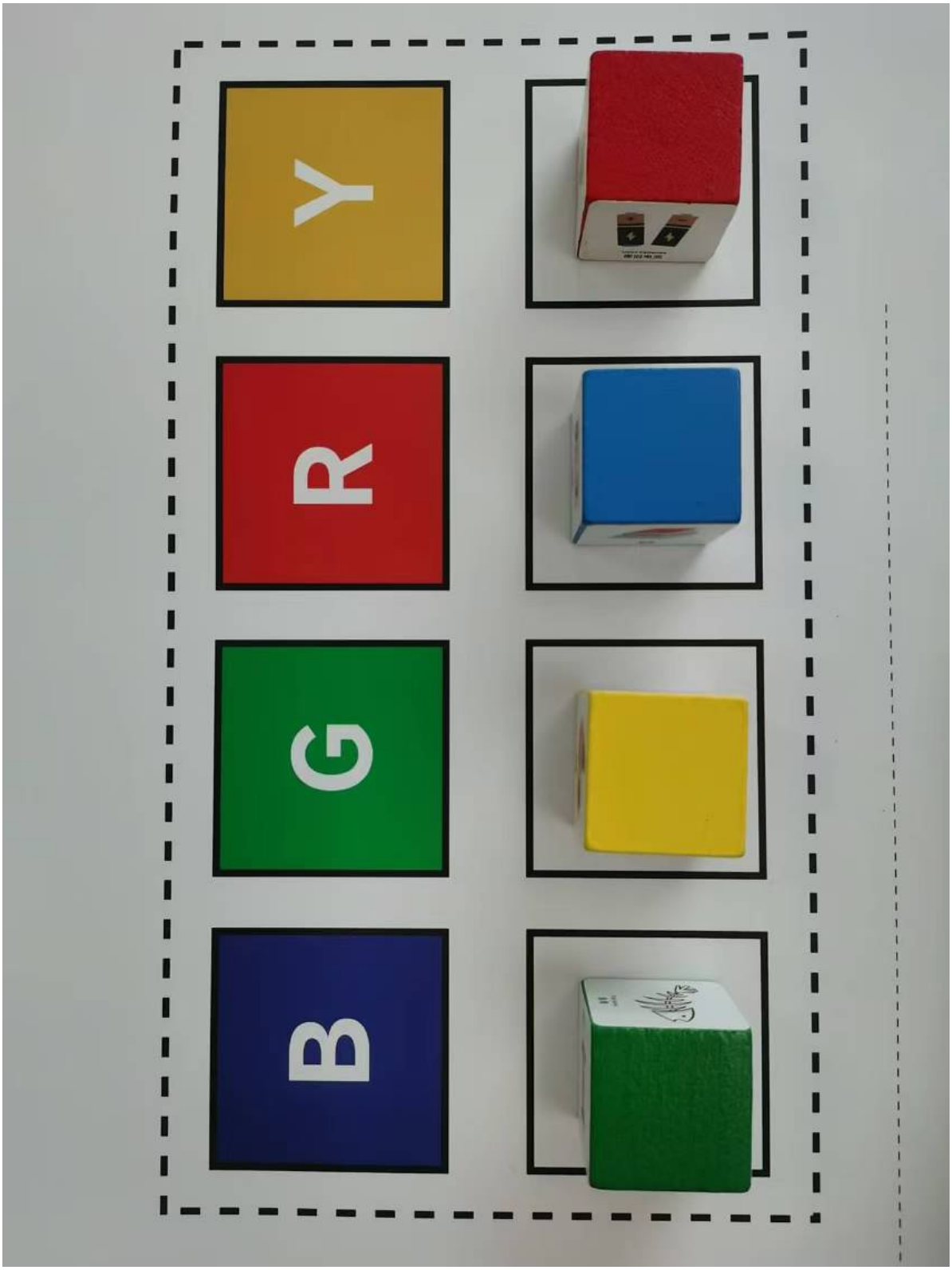
## 1. Gesture recognition instructions

The gesture stacking function mainly combines the mediapipe gesture recognition function, the robot arm and the map gameplay. A total of five gestures are recognized, numbers 1-5, where numbers 1-4 represent the clamping positions 1-4 on the map, and number 5 means clearing the data and restoring the default stacking layer number.

## 2. Map placement

Place the blocks at positions numbered 1-4, and the colors can be chosen randomly.


B
G
R
Y
1
2
3
4




## 3. About code

Code path：~/jetcobot_ws/src/jetcobot_grasp/scripts/1_gesture_recognition_stacking.py

~/jetcobot_ws/src/jetcobot_utils/src/jetcobot_utils/grasp_controller.py

Control the movement and grasping functions of the robotic arm.

```
def ctrl_arm_move(self, index):
        if index >= 5:
            self.graspController.ctrl_nod()
```

```
            return
        self.graspController.goColorOverPose()
        if index == 1:
            self.graspController.goApriltag1fixedPose(2)
        elif index == 2:
            self.graspController.goApriltag2fixedPose(2)
        elif index == 3:
            self.graspController.goApriltag3fixedPose(2)
        elif index == 4:
            self.graspController.goApriltag4fixedPose(2)
        time.sleep(1)
        self.graspController.drop_gripper(1)
        self.graspController.close_gripper(1)
        self.graspController.ctrl_gripper_height(200, 1.5)
        # self.graspController.goColorOverPose()
        # time.sleep(1)
        self.graspController.goStackingOverPose()
        time.sleep(1)
        self.block_num = self.block_num + 1
        self.graspController.goStackingPose(str(self.block_num))
        time.sleep(1)
        self.graspController.open_gripper(1)
        self.graspController.ctrl_gripper_height(150+self.block_num*30, 1.5)
        self.graspController.init_pose()
        time.sleep(.5)
```

The position coordinates corresponding to numbers 1-4.

If the clamping position coordinates are inaccurate, you can modify this coordinate value appropriately.

```
# Apriltag Label location
    def goApriltag1fixedPose(self, layer=1):
        coords = [-60, 170, 105+50*(layer-1), -175, 0, -45]
        self.go_coords(coords, 2)

    def goApriltag2fixedPose(self, layer=1):
        coords = [10, 170, 105+50*(layer-1), -175, 0, -45]
        self.go_coords(coords, 2)

    def goApriltag3fixedPose(self, layer=1):
        coords = [75, 170, 105+50*(layer-1), -175, 0, -45]
        self.go_coords(coords, 2)

    def goApriltag4fixedPose(self, layer=1):
        coords = [140, 170, 110+50*(layer-1), -175, 0, -45]
        self.go_coords(coords, 2)
```

The coordinate value of the stacking position.

If the stacking position coordinate is inaccurate, you can modify this coordinate value appropriately.

```
# stacking
```

```
    def goStackingNum1Pose(self):
        coords = [140, -160, 110, -180, -2, -43]
        self.go_coords(coords, 3)

    def goStackingNum2Pose(self):
        coords = [145, -160, 145, -180, -2, -43]
        self.go_coords(coords, 3)

    def goStackingNum3Pose(self):
        coords = [145, -160, 175, -180, -2, -43]
        self.go_coords(coords, 3)

    def goStackingNum4Pose(self):
        coords = [145, -160, 205, -180, -2, -43]
        self.go_coords(coords, 3)
```

# 4. Start code

**Start roscore**

- If you are using Jetson Orin NX/Jetson Orin Nano board. You need to enter the Docker environment using the following command.
- Then, run roscore

```
sh ~/start_docker.sh
roscore
```

- If you are using Jetson  Nano board. You need to enter the following command directly.

```
roscore
```

**Start the program**

Re-open a terminal.

- If you are using Jetson Orin NX/Jetson Orin Nano board. You need to enter the Docker environment using the following command.

```
sh ~/start_docker.sh
rosrun jetcobot_grasp 1_gesture_recognition_stacking.py
```

- If you are using Jetson  Nano board. You need to enter the following command directly.

```
rosrun jetcobot_grasp 1_gesture_recognition_stacking.py
```

# 5. Experimental results

If the set gesture action is recognized, the robot arm will perform the corresponding action.

When the robot arm recognizes the number for the first time, it will go to the corresponding number position to pick up the building block and put it on the first layer;

When it recognizes the gesture number for the second time, it will go to the corresponding number position to pick up the building block and put it on the second layer;

When it recognizes the gesture number for the third time, it will go to the corresponding number position to pick up the building block and put it on the third layer;

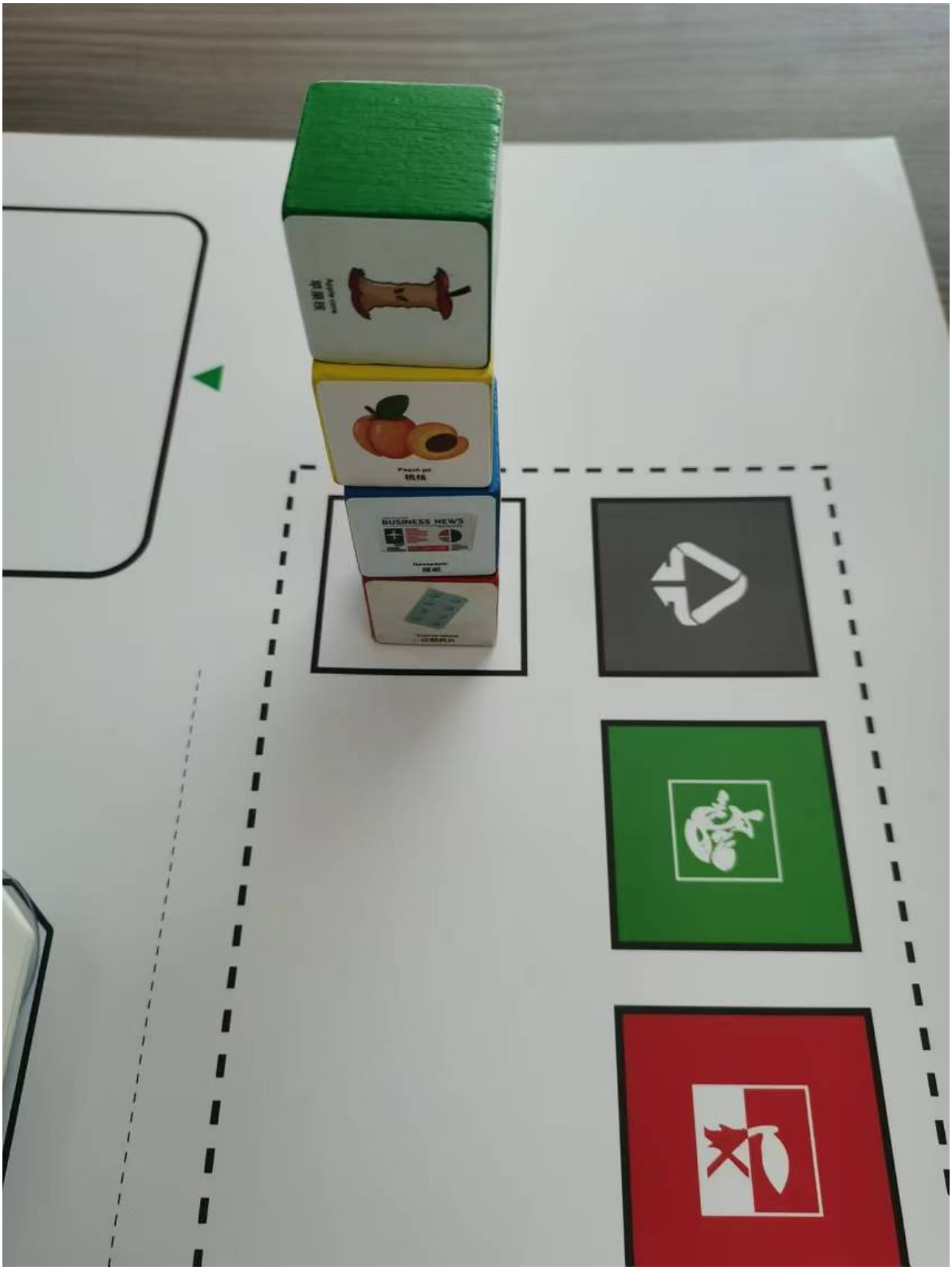
When it recognizes the gesture number for the fourth time, it will go to the corresponding number position to pick up the building block and put it on the fourth layer;

Each number can only be recognized once, and the action will not be executed if it is recognized multiple times;

When the number five is recognized, it will nod, then clear the record and start again.

The gestures and actions correspond to the following:

| **Gestures** | **Function** |
| --- | --- |
| Gesture_1 | Go to position 1 and pick up the building blocks and stack them up |
| Gesture_2 | Go to position 2 and pick up the building blocks and stack them up |
| Gesture_3 | Go to position 3 and pick up the building blocks and stack them up |
| Gesture_4 | Go to position 4 and pick up the building blocks and stack them up |
| Gesture_5 | Clear the record |


Peach pit
桃核
BUSINESS NEWS