电子科技大学
格拉斯哥学院
Glasgow College, UESTC

Team Design Project and Skills (2019-20)

Final Report

2019-2020 Group-16 TDPS M.J.D

Weixi Xiong	2017200603001	2357797X
Yijun Zhao	2017200604030	2357836Z
Yimin Tian	2017200603010	2357806T
Qingyun Wang	2017200603015	2357811W
Jingyuan Feng	2017200501036	2357510F
Yuxiao Luo	2017200501025	2357499I
Xingyue Wu	2017200503007	2357561W
Zhuozhao Liu	2017200503026	2357580L
Wentao Du	2017200603017	2357813D
Mianzhe Wu	2016200304028	2289329W

1. ABSTRACT[1]

Team Design project and Skills (2019-20) will be conducted online as the epidemic disease. The project is simplified from 6 tasks into 5 tasks, and the software ***WEBOT*** is used for simulation. We are supposed to build our simulation environment, assemble the motor and write the controller to complete series of special tasks in patio in a team work manner. We used distance sensor to detect and follow the line on the ground, mechanical arm and color detector to recognize the yellow mark and release fish food, another distance sensor to detect bridge and avoid obstacles. When it comes to patio2, distance sensor is also used to detect the arch and send message to motors of wheels, and color detector & line follow mode is used for selecting different routes. Our group initially regrouped into 2 groups to build the physical environment and learning APIs relatively. When the preparations were completed, it is discovered that two groups need integration to take responsibilities of each 5 tasks. By the end of May 2020, all 5 groups finished their tasks and we reached the integration and preparation of presentation. The integration is achieved in order of the task in a logic order, and the same as presentation. We finally achieved the perfectly simulation thanks to the contributions from all group members. It is found from this project that the personal investigation and cooperation are both more than significant and necessary in practical team work especially in the condition of such a complex project. Also, the in-time adjustment of the allocation and macroscopical grasp of the progress play an important role. The willing to perfectly complete our own work and help each other at the same time is the key to achieve the victory of the whole project.

[1] Weixi Xiong 2357797X 2017200603001

2. TABLE OF CONTENTS[2]

[2] Weixi Xiong 2357797X 2017200603001

3. LIST OF FIGURES AND TABLES[3]

FIGURES

[3] Weixi Xiong 2357797X 2017200603001

TABLES

4.ACKNOWLEDGEMENTS[4]

First of all, we would genuinely appreciate the kind and generous contribution and guidance from our professors Dr. Wasim Ahmad (Course Coordinator) and Dr. Abdullah Al-Khalidi most. Professors contribute themselves into teaching us how to solve all tasks, how to cooperate and how to finish the whole project. We are also obliged to other teams for their unreserved help and precious learning experience.

Thanks to all team members in group-16 as well, it is the contribution from everyone makes our project a successful cooperation. Please allow me show my deepest recognition and gratitude to them.

Personal Contribution List

Name	Project Allocation	Detailed Contribution
Qingyun Wang	Task1	Infrared Sensor, Line Following Technique and Physical Robot Structure
Weixi Xiong	Task2	Integration and Main Writing of final report, Mechanical arm, Arrangement and integration of Notebooks, Patio construction
Yijun Zhao	Task2	Color Detecting & Combine the Code of Task-1, 2 and 3
Zhuozhao Liu	Task3	Testing Model for Distance Sensors, Trees Avoidance and Adjustment
Jingyuan Feng	Task3	Cars bridge crossing, Code Combination， Report Time Arrangement, Environment Construction
Xingyue Wu	Task4	Selection of Sensors and Basic programming
Yuxiao Luo	Task4	Sensor Selection, Writing Code, Combine Task-3 and Task-4
Yimin Tian	Task5	Design Route Map for Task-5, Develop Module of Color Box Recognition, Optimize Module of the Colored Line Follow
Wentao Du	Task5	Patio construction, Color Verification, Solutions for Task-5
Mianzhe Wu	Task5	Line following, Change Directions, Color Recognition

[4] Weixi Xiong 2357797X 2017200603001

5.Introduction

Generally, the project is divided into 5 tasks on 2 patios. Goal functions includes line follow, mechanical arm & color detection, avoiding obstacles & crossing bridge, crossing arch and color detection & colored line follow. The entire project is supposed to use one motor which can only load one controller. Thus, what we need to build is a robot that can follow the trace, detect and interact with color, avoiding obstacles, holding fish food and detecting distance. The robot and tasks can be described as follows:

5.1 LINE FOLLOWING MODE[5]

This task is a line-following task. It requires the robot to follow line on the ground and move from the start to the end. I analyzed the basic information about ***WEBOT*** by the tutorials and samples in the ***WEBOT*** guide tour. By comparing the scene tree nodes and the controller codes of the robot, I chose the sample E-puck line-demo as my main reference of my task.

5.2 MECHANICAL ARM & COLOR DETECTION[6]

To guarantee the robot can hold the fish food and release the food in the pond when yellow is detected, the mechanical arm needs to be controlled by a function that can be nested in the ***if*** function used by color detector. In ***WEBOT*** world, the mechanical arm is a special and individual ***robot*** node nested in the motor, so it has an individual controller. ***KUKA's youBot MOBILE ROBOTIC*** is a proto whose 'arm has five degrees of freedom and a linear gripper.' [1] It is selected as the gripper of the robot.

The color detection part asks us to use a camera to get the real time image and when it finds the yellow box, the car will operate the relevant code accompanied with the mechanical arm.

5.3 AVOIDING OBSTACLES & CROSSING BRIDGE[7]

Task-3 main task is to turn the car after the detected object, mainly divided into two parts, one part is to make the car rotates the right angle after the bridge is detected to make sure it can go through the bridge safely, the other part is to make the car can

[5] QingYun Wang 2357811W 2017200603015
[6] Weixi Xiong 2357797X 2017200603001 & Yijun Zhao 2357836Z 2017200604030
[7] Zhuozhao Liu 2357580L 2017200503026 & Jingyuan Feng 2357510F 2017200501036

rotate right angle in front of the trees to ensure that the car can go to the arch of task-4. After our study and discussion, we finally decided to use the distance sensor to accomplish this task.

5.4 CROSSING ARCH & INTEGRATION[8]

The aim of Task 4 is to find the arch and go through it and then follow the line on the ground. After passing through the arch, you should turn right and go straight without colliding with the trees. It can be divided into 2 parts: detect the arch and turn 90 degrees.

5.5 COLOR DETECTION & COLORED LINE FOLLOW[9]

Task 5 is to navigate car to the assigned colored line according to the color of box and execute line following procedures to reach the final line. In the development, task 5 is divided into several small parts, designing route lines, recognizing colored box and following colored lines. As task 5 shares line following components with task 1, one more hardware we need to add to the project 4-wheel car is camera module placed in the front of the car.

[8] Xingyue Wu 2357561W 2017200503007 & Yuxiao Luo

[9] Yimin Tian 2357806T 2017200603010 & Wentao Du 2357813D 2017200603017 & Mianzhe Wu 2289329w 2016200304028

6.Overall System Design Approach[10]

The whole project is divided into 5 tasks, so we generally plan to regroup 10 members into 5 teams to complete each task then straight into integration.

6.1 INITIAL DESIGN AND WORK ALLOCATION

Yuxiao Luo is voted to be our group leader at the very beginning of the project, and the note arrangement and integration are responsible by Weixi Xiong. Jingyuan Feng are mainly connecting with professors and submitting materials. Programming language is decided to ***C#***.

6.1.1 WORK ALLOCATION

For initializing the project, the physics environment needs to be built and the protos, APIS need to be learned. According to this, regrouping result is as follow:

6.1.1.1 Group-1 Building Environment

Group members: Weixi Xiong, Yuxiao Luo, Jingyuan Feng, Wentao Du, Yimin Tian

This group is mainly responsible for building the physical environment, including pond for task-1 and task-2, bridge, trees and arch for task-3 and task-4, color box for task-5 and lines on the floor for line detection. The practical construction is completed mainly by Yuxiao Luo, Weixi Xiong and Jingyuan Feng, adjustment is conducted by Yimin Tian, note-taking is a job for Wentao Du.

6.1.1.2 Group-2 API Learning

Group members: Qingyun Wang, Yijun Zhao, Zhuozhao Liu, Xingyue Wu, Mianzhe Wu

This group is working on tutorials and other learning materials from *http://www.cyberbotics.com/.* The initial solutions of each tasks need to be thought as well. Thanks to arrangement from Yijun Zhao, Qingyun Wang is responsible for task-

[10] Weixi Xiong 2357797X 2017200603001

1, Yijun Zhao for task-2, Zhuozhao Liu for task-3, Xingyue Wu for task-4 and Mianzhe Wu for task-5.

6.1.2 INITIAL DESIGN

The patio is built and mostly using ***solid*** nodes in ***WEBOT***. The nodes list and conceptual graph is shown below:

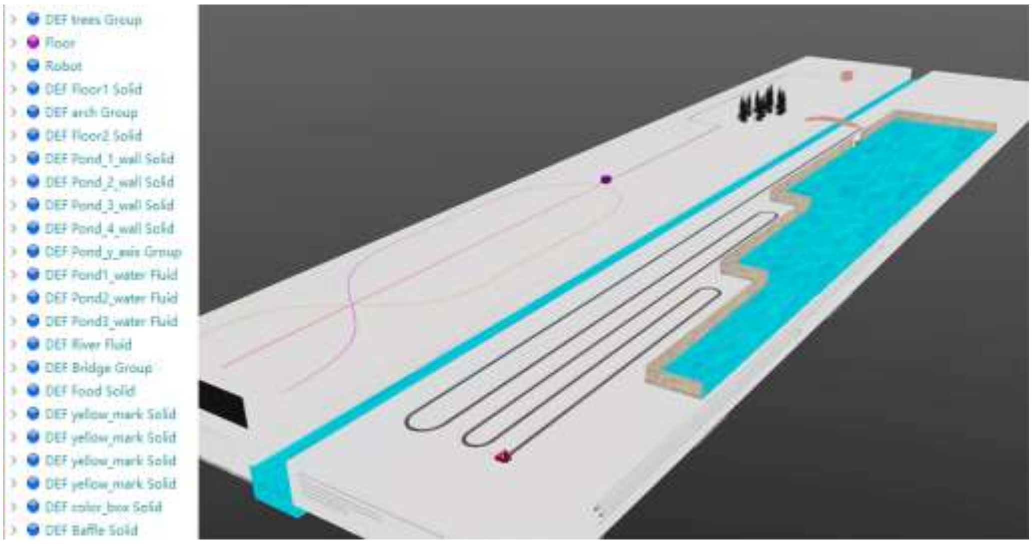


Figure1 - Conceptual Graph of Patio and Nodes List

With a ***floor*** holding the bottom, 2 patios are built using ***solid*** node. On the ground, pond is a combination of 3 ***liquid*** nodes and walls are surrounding the water. The bridge and arch are both built by 3 ***solid-shape-box*** nodes. The trees are using the proto from ***WEBOT***. The ***boundobject*** and ***physics*** are all set.

After achievement of the physical world, API learning group can conduct practical test. Then the task-goal design can be on schedule.

6.2 TASK DIVISION AND INTEGRATION

To finish 5 tasks, we reached a consensus that the Building Environment group members participate in API Learning groups to work on each task by cooperation.

6.2.1 TASK DIVISION

Based on discussion and assessment of 10 members, collision is regard to be easily occur in integration of task-3 and task-4. Also, task-1 is suitable for one person while task-5 needs work of 3 members. Then the regrouping result came out:

<table>
<tr><td colspan="2">Task-1</td><td>Line Follow</td><td>Qingyun Wang</td></tr>
<tr><td colspan="2">Task-2</td><td>Mechanical Arm & Color Detection</td><td>Weixi Xiong, Yijun Zhao</td></tr>
<tr><td rowspan="2">Task-3&4</td><td>Task-3</td><td>Avoiding Obstacles & Crossing Bridge</td><td>Zhuozhao Liu, Jingyuan Feng</td></tr>
<tr><td>Task-4</td><td>Crossing Arch & Integration</td><td>Xingyue Wu, Yuxiao Luo</td></tr>
<tr><td colspan="2">Task-5</td><td>Color Detection & Colored Line Follow</td><td>Wentao Du, Yimin Tian, Mianzhe Wu</td></tr>
</table>

Table1 – Mainly Used Functions and Corresponding Order

All groups are supposed to finish work up to the middle of May to guarantee integration and presentation can be prepared.

6.2.2 INTEGRATION

The integration work is advanced in order of the task. For instance, once task-1 is completed, the code and world are supposed to send to task-2 group immediately. The next group need to add their technical results to previous work. When problems occur, seeking help from other group members are permitted. Practically, integration of task-2 and task-1, task-3 and task-4, task-2 and task-5 came out from the endeavor of bilateral members.

7. Technical Content

7.1 COMPLETION OF TASK-1

NAME: QINGYUN WANG

7.1.1 PHYSICAL WORLD

After my teammates completing the whole patio, firstly, I need to make the line map. I used PowerPoint and Photoshop to make the picture of line map and adapt its length and wideness to fit the environment.

Figure2 – Line Map

Then I made the basic robot by referring the tutorial 4-wheel-robot. For details, the body of my robot is a box (0.5*0.2*0.5). The height of wheel is 0.05 and radius is 0.08.

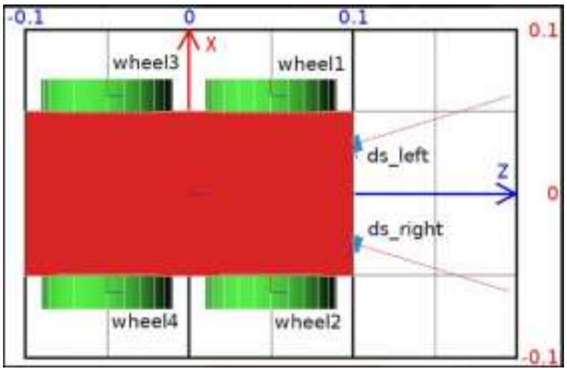


Figure3 - Top View of the 4 Wheels Robot

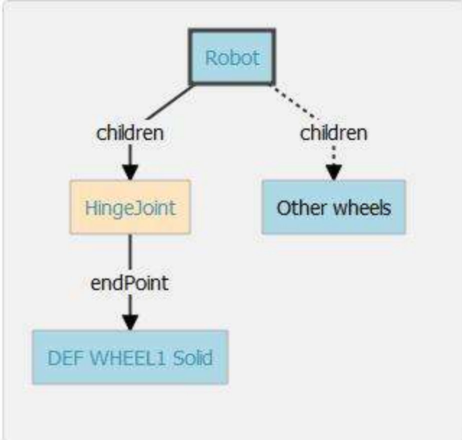


Figure4 - Tree Nodes of the 4 Wheels Robot

To achieve line following, I used the ground sensor of E-puck robot. It contains three small infra-red sensors. As shown below, (ignore the white object) the big green board is the sensor circuit, which does not have real function in this program. These three black boxes ***G0, G1*** and ***G2***(circled by blue) are infra sensors.

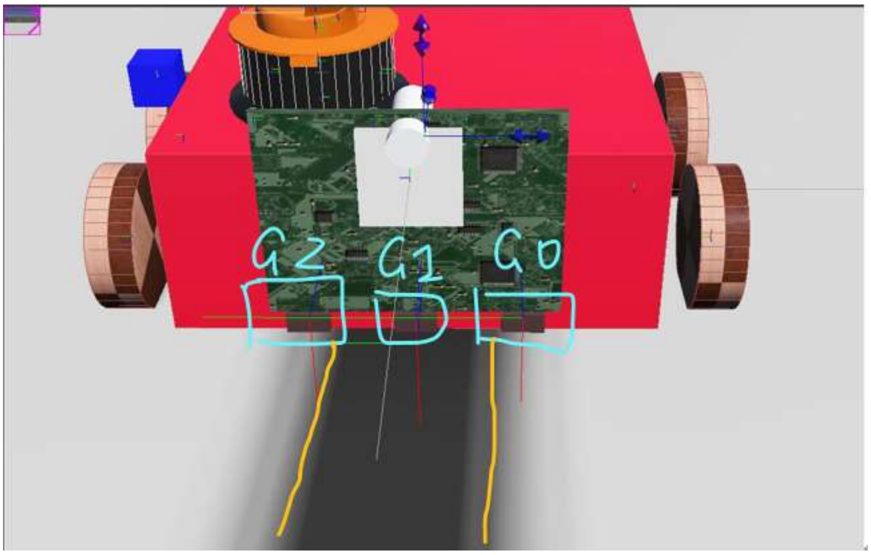


Figure5 – The Ground Sensors

7.1.2 INFRARED WHITE-BLACK RECOGNITION

Infrared sensors are photosensitive devices that can convert infrared radiation energy into electrical energy. The infrared sensor is designed by the circuit, when the infrared light intensity is received, the circuit is turned on, and the value collected on the microcomputer is small. When the infrared light is weak, the circuit is not turned on, and the value collected by the microcomputer is large.

Using this principle, black absorbs the light from the infrared emitting tube, so the infrared received by the infrared receiving tube is weak. The sensor circuit is not conductive, and the feedback value of black is large. For the white part, the infrared received by the infrared receiver tube is strong. The sensor circuit is turned on, and the feedback value of white is small. Based on this principle, the robot distinguishes black and white lines.

Thus, in my program, as shown in figure-4, the rays of ***G0*** and ***G2*** are exactly on the edge of black line (on the boundary of black and white). As mentioned before, the sensor can get different back values for white and black. What I need is the gap value Deltas. In the code below (in figure-5), that it is ***Deltas= G2-G0***. Then I set two parameters ***LFM_FORWARD_SPEED, LFM_K_GS_SPEED*** (in figure 7) and import those into an algorithm (in figure-6). In this way, I get the left wheel and right wheel velocity by the gap of feedback value of left and right infrared sensors.

```
void LineFollowingModule(void) {
  int DeltaS = gs_value[GS_RIGHT] - gs_value[GS_LEFT];

  lfm_speed[LEFT] = LFM_FORWARD_SPEED - LFM_K_GS_SPEED *DeltaS;
  lfm_speed[RIGHT] = LFM_FORWARD_SPEED + LFM_K_GS_SPEED *DeltaS;
}
```

Figure6 – The Algorithm for Velocity

```
#define LFM_FORWARD_SPEED 3000
#define LFM_K_GS_SPEED 3.5
```

Figure7 – The Parameters

In addition, the parameter ***LFM_FORWARD_SPEED*** is the initial velocity and forward velocity, and ***LFM_K_GS_SPEED*** is a coefficient controlling velocity difference of left and right wheels.

In the real world, in briefly, when the robot is about to deviate, left wheel and right wheel will adapt their relative velocity and reorient by the feedback value of ground sensor.

7.2.1 MECHANICAL ARM AND OTHER DISTRIBUTION

NAME: WEIXI XIONG

I have been mainly responsible for the mechanical arm part in task-2, target for assembling the arm and food box onto the car and compiling the controlling code of mechanical arm into a callable function. Also, I have participated in code writing of task-2 with Yijun Zhao and the combining 5 tasks codes.

In other fields, the notebook of the whole team is collected, reorganized, and submitted by me. Including arranging team members to take group notes in order and establishing a text file using cloud storage.

In addition, about half of the construction of the physical environment (the patio) is also come from my work. And almost half of the main part of group work in final report is completed by me. I also participated the integration of the all 5 tasks.

Details and investigations will be discussed in the Analysis and Discussion part.

7.2.2 COLOR DETECTOR AND OTHER CONTRIBUTIONS

NAME: ZHAO YIJUN

I am getting in charge the task 2 with Weixi Xiong. I am mainly responsible for the camera as well as the color detecting and combine the task 2 code with the task 1 code. So, I also work with Qingyun Wang who is responsible for task 1 and help to connect task 2 and task 3 together.

What's more, I arranged the division of work of the API team.

7.3.1 TREES AVOIDANCE

NAME: ZHUOZHAO LIU

My personal contribution includes the things I introduce in next section. What's more, my personal work includes building a simulation car model and use it to test task3, by modifying the codes and the distance sensor, I use a while loop to connect passing the bridge and avoid the trees. By discussing the code with team members to achieve our final goals, we do a lot of adjustment.

Building the car model for simulation, writing codes, testing the car avoidance and combing the crossing the bridge and trees avoidance codes together as well as discussion with others to improve.

7.3.2 TECHNICAL CONTENT

NAME: JINGYUAN FENG

In this experiment, I was mainly responsible for the bridge crossing part of task-3. The goal was to make the car turn correctly and pass the bridge safely after it detected the bridge. I also worked with Zhuozhao Liu to improve the task3 code and I participated in the combination of the whole five tasks.

Mainly responsible for the bridge crossing part of task3 and also worked with team to improve and combine the code of task 3 and the whole experiment. In other fields, the time of each meeting of our group was inquired and submitted by me. I took notes of the group once and participated in the construction of the scene.

7.4.1 TECHNICAL CONTENT

NAME: YUXIAO LUO

The task which I am responsible for is task 4, turning right at the arch. My contribution is considered the fundamental solution model and write the code. The model which I use is distance sensor, which I put on the left side of the car.

When the car goes straight, the sensor will detect the arch on it left, and turn for 90 degree through controlling the code. My next contribution is the writing the controller of the distance sensor. It is achieved by a while loop and three if statements, when the sensor detect the arch, it will execute different codes, and will turn left or turn right in the bridge, trees, and the arch.

Meanwhile, my next contribution is combining task 3 and task 4.

Figure8 – Issues in Task-3 and Task-4

There are some problems in task-3 and task-4, first is that the car will turn left after passing the tree because the left sensor will detect the tree again.

Figure9 – Crossing Arch Process

So that I optimized the program, then just it just needs one loop for the two tasks and four counters to record the positions and controlling the swerving degree.

```
bool avoid_obstacle_counter1 = 0;
bool avoid_obstacle_counter2 = 0;
bool avoid_obstacle_counter3 = 0;
bool avoid_obstacle_counter4 = 0;
```

Figure10 – Code for Task-4 1

```
while (wb_robot_step(TIME_STEP) != -1) {
  double left_speed = 5;
  double right_speed = 5;
  if (avoid_obstacle_counter1 > 0&& avoid_obstacle_counter3 == 0&&avoid_obstacle_counter4==0)
   {
    avoid_obstacle_counter1--;
    left_speed = -2.1;
    right_speed = 2.1;
    if(avoid_obstacle_counter1==0)
    {avoid_obstacle_counter4=1; }}
  else if ((avoid_obstacle_counter1 > 0)&&(avoid_obstacle_counter4 ==1)) {
    avoid_obstacle_counter1--;
    left_speed = -2.1;
    right_speed = 2.1;
    if(avoid_obstacle_counter1 ==0)
    break;
  }

 else if (avoid_obstacle_counter2 > 0) {
    avoid_obstacle_counter2--;
    left_speed = 2.1;
    right_speed = -2.1;
    avoid_obstacle_counter3=1;
    if(avoid_obstacle_counter2==0)
    {left_speed = 5;
    right_speed = 5;
    wb_motor_set_velocity(wheels[0], left_speed);
    wb_motor_set_velocity(wheels[1], right_speed);
    wb_motor_set_velocity(wheels[2], left_speed);
    wb_motor_set_velocity(wheels[3], right_speed);
    passive_wait(6);
    avoid_obstacle_counter3=0;
    }
  }
   else{ // read sensors
    double ds_values1[1];
    for (i = 0; i < 1; i++)
      ds_values1[i] = wb_distance_sensor_get_value(ds[i]);
    if (ds_values1[0] <999.0 )
      {passive_wait(1);
      avoid_obstacle_counter1 = 107;}
    double ds_values2[1];
    for (i = 0; i < 1; i++)
      ds_values2[i] = wb_distance_sensor_get_value(ds_r[i]);
    if (ds_values2[0] <999.0 )
      avoid_obstacle_counter2 = 117 ;
```

Figure11 – Code for Task-4 2

My solution of distance sensor is also used in task 3.

7.4.2 TECHNICAL CONTENT

NAME: XINGYUE WU

I am mainly responsible for basic design and further help of task-4 required for the robot to find and go across the arch. In preparation, it is my duty to make decisions about sensors first, like selecting the type, deciding the position, calling the sensors, etc. Before the combination the whole tasks, I test all tutorials by myself and finally find some useful and helpful content for our task.

In addition, I assist my team. Researching and sharing tutorial videos and codes, and the code of task-3 and task-4 are based on the primary code of avoiding obstacles. Helping my partner Yuxiao Luo to change and test the code, and record the process and final results in our notebook.

7.5.1 TECHNICAL CONTENT

NAME: WENTAO DU

Firstly, I was a member of environment construction group and I took part in the construction of the map the car drive on. Then I worked for task-5. At this part, I find some basic codes for camera and distance sensor in the library of ***WEBOT*** as well as mastering the skill of controlling them and shared these with my partner for task-5. During the meeting in task-5 group, I came up with two ideas to finish our task and shared them with member who was responsible for task-2. To choose the better solution I build a robot with camera and distance sensor to test them. Also, I found we can use a module defining the speed of wheels at different sides to control the car to turn when it finds the color box, and this module is also used in task-2. Finally, I helped my partner write and optimize the code for task-5.

7.5.2 TECHNICAL CONTENT

NAME: YIMIN TIAN

In task-5, I am responsible for designing route maps for task-5, developing colored box recognition module and optimizing colored line following module. In detail, route maps implemented by ***Microsoft Paint 3D***; colored box recognition developed with

Camera node RGB color space; colored line following module is optimized by adjusting initial speed parameter and steering parameter.

7.5.3 TECHNICAL CONTENT

NAME: MIANZHE WU

I am mainly responsible for color recognition and line following in task-5. In this task the car needs to recognize red, yellow and blue then three lines need to be set for the car to follow. These are accomplished by my partner, then I am in charge of adjusting the direction of car and follow lines.

8. Analysis and Discussion

8.1 STABILITY OF ROBOT

NAME: QINGYUN WANG

When the robot does line following, there are some practical factors that impacts the stability of robot. In general, I solve the deviation problem in three approaches.

The first one is to set a suitable speed of robot. I used the Control Panel Variable Method to modify two parameters and increase the speed of robot while ensuring stability.

Secondly, the mass of robot tightly impacts stability, because mass is proportional to inertia and robot may lose control at the semi-circle corner if the inertia is too large. Thus, I decreased the density of robot and wheels to 100 and 1, respectively.

Figure12 – The Semi-Circle Corner

Thirdly, I increase the distance between left and right wheels (in figure 10). It enhances stability of robot in the physical structure of robot.

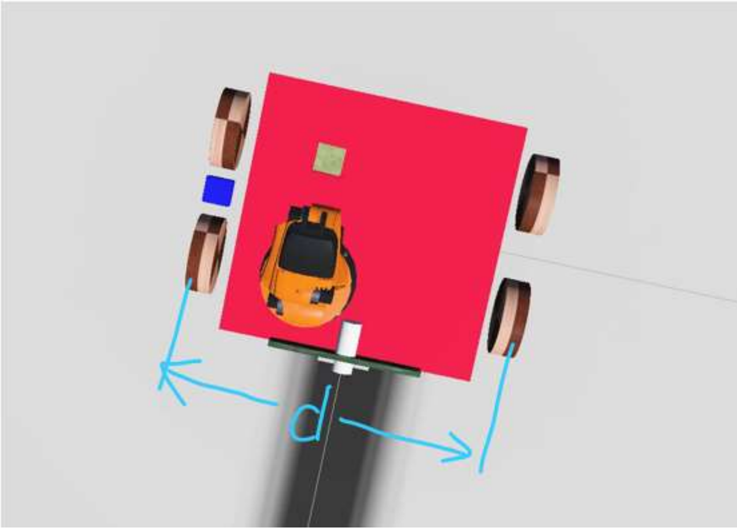


Figure13 – Distance Between Left and Right Wheels

8.2.1 ANALYSIS OF MECHANICAL ARM

NAME: WEIXI XIONG

My task is adjusting the mechanical arms and writing controller for it. Therefore, this part will be separated into 2 parts.

PHYSICAL ADJUSTMENT

The robot arm is decided to ***KUKA's youBot MOBILE ROBOTIC*** whose conceptual graph is shown below:

Figure14 - Conceptual Graph For youBot

The youBot is a mobile robotic arm developed by KUKA. Its arm has five degrees of freedom and a linear gripper. Its base has four Mecanum wheels allowing for omnidirectional movement. These wheels are efficiently modeled using asymmetric friction.[11]

In details, part ① is a motor carrying four sensors and four wheels used to hold the arm. In our case, as we have our own motor so this part can be deleted. Part ② is a pedestal with a hinge that can turn 360 degree. What is important is that the arm can only reach the platform parallel to the top of the pedestal. So that the height is redundant in our task and is adjusted. Part ③④⑤ are hinges in vertical direction which are mainly working in our project. Their turning can be controlled by ***wb_set_position()***, which will be discussed in next part. Part ⑥ is also a hinge that can turn like the one in ② does, used to adjust the direction of the object. ⑦ is simply a gripper. To remind, 7 sensors on arms are also provided by ***KUKA's youBot MOBILE ROBOTIC***, but not used in our project. Degree of freedom of all of the hinges are tested and list in table:

Hinge		②	③	④	⑤	⑥
Degree of Freedom (radian)	**Maximum Position**	1.638π	π	1.416π	0.989π	1.622π
	Minimum Position	-1.638π	-0.627π	1.467π	-0.989π	-1.622π

Table2 – Degree of Freedom of Hinges of youBot

[11] http://www.cyberbotics.com/doc/guide/youbot

To set the arm and the food box onto the car, I used 4 screws to fix the arm at the north east corner of the top of the box, and built 4 baffles to make sure food box would not fall in task-1. The screenshot is provided:

Figure15 – Overall View

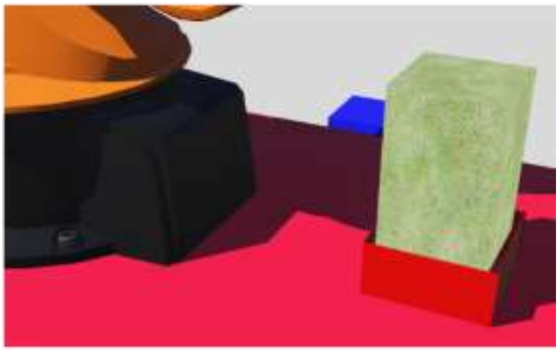


Figure16 – Screws and Baffles

By testing the physical interaction in WEBOT, I found that when two objects both with ***bounding object*** overlap, the motion and interaction will become more than unstable. So the distance between the car and the arm is adjusted to 1mm. And with the help of the screws, the arm is fixed. The ***screw radius*** is 0.003m and ***screw length*** is 0.02m using a proto from WEBOT. Then the baffle is using ***BOX*** with size of 0.0001, 0.06, 0.035. All data is controlled to prevent to be too short to lead to food falling from the car and too high to lead to gripper being unable to grasp the object and turn.

The ***physics*** of all hinges and components are set as low as possible as line-follow in task-1 will be affected when weight is too large.

CODE AND DISCUSSION

Firstly, the controller of the arm is individual from the controller of the whole robot as it is a ***ROBOT*** node embedded into the motor. The two robots have no interaction which means the sensors on the car can not be invoked by arm-controller. So a special color-detector only for arm need to be added onto the arm, but in our case we found that let the car ***wait*** for the time of task-1 is sufficient as the simulation is ideal. Then the code will be discussed by 3 parts.

The first one is the headers. Totally 3 extra headers are included which are ***<arm.h>, <gripper.h>, <math.h>***. All are default headers from ***KUKA's youBot MOBILE ROBOTIC*** and is analyzed. ***<arm.h>*** defined series of ***set_position()*** functions for initial action which are placed in the front of the main function to set the initial condition for arm including ***ARM_FRONT_CARDBOARD_BOX, ARM_FRONT_LEFT*** and so on. All 'position' functions are used in ***automatic_behavior(),*** which is preinstalled. What are used in main function is literally 4 functions which are listed below:

Function Name	**Order**
arm_increase_height()	*current_height++*
arm_decrease_height()	*current_height--*
arm_increase_orientation()	*current_orientation++*
arm_decrease_orientation()	*current_orientation--*

Table3 – Mainly Used Functions and Corresponding Order

All 4 have self-error-correction mode to check whether exceed the degree of freedom.

<gripper.h> simply provide ***gripper_grip() and gripper_release()*** to control the gripper. As a reminder, maximum position is 0.025m and minimum position is 0m. ***<math.h>*** provide formula like ternary operator and matrix operator.

In addition, the MAKEFILE need some extra code to supply headers:

```
WEBOTS_HOME_PATH=$(subst $(space),\ ,$(strip $(subst \,/,$(WEBOTS_HOME))))
RESOURCES_PATH = $(WEBOTS_HOME)/projects/robots/kuka/youbot
INCLUDE = -I"$(RESOURCES_PATH)/libraries/youbot_control/include"
LIBRARIES = -L"$(RESOURCES_PATH)/libraries/youbot_control" -lyoubot_control

### Do not modify: this includes Webots global Makefile.include
include $(WEBOTS_HOME_PATH)/resources/Makefile.include
```

Figure17 – Extra code in MAKEFILE

Second part is some functions. A ***passive_wait()*** function is added to replace the ***wait()*** function in ***C#*** but not accessible in WEBOT, and a setting initial position function called ***automataic_behavior()*** which is mentioned before. The code is provided, and is necessary before any order in main function.

```
static void automatic_behavior() {
  passive_wait(2.0);
  gripper_release();
  arm_set_height(ARM_FRONT_CARDBOARD_BOX);
  passive_wait(4.0);
  gripper_grip();
  passive_wait(1.0);
  arm_set_height(ARM_BACK_PLATE_LOW);
  passive_wait(3.0);
  gripper_release();
  passive_wait(1.0);
  arm_reset();
  passive_wait(5.0);
  gripper_grip();
  passive_wait(1.0);
  passive_wait(1.0);
  gripper_release();
  arm_set_height(ARM_BACK_PLATE_LOW);
  passive_wait(3.0);
  gripper_grip();
  passive_wait(1.0);
  arm_set_height(ARM_RESET);
  passive_wait(2.0);
  arm_set_height(ARM_FRONT_PLATE);
  arm_set_orientation(ARM_RIGHT);
  passive_wait(4.0);
  arm_set_height(ARM_FRONT_FLOOR);
  passive_wait(2.0);
  gripper_release();
  passive_wait(1.0);
  arm_set_height(ARM_FRONT_PLATE);
  passive_wait(2.0);
  arm_set_height(ARM_RESET);
  passive_wait(2.0);
  arm_reset();
  gripper_grip();
  passive_wait(2.0);
}
```

Figure18 – Code for automatic_behavior()

Last part is the main function. The main function first use a ***passive_wait(120.0)***, in which 120 is the time for task-1 and can be replaced by an ***if()*** sequence in practice once the special color detector is added. Then a series of actions are calculated, tested and added to the main function. To avoid collision between orders, ***passive_wait()*** is also used to help. The whole code is shown below with annotation,

```c

int main(int argc, char **argv) {
  wb_robot_init();
  passive_wait(120.0);//The time of task 1
  arm_init();
  gripper_init();
  passive_wait(2.0);

  if (argc > 1 && strcmp(argv[1], "de mo") == 0)
   automatic_behavior();//set the initial position of hinges

  passive_wait(1.0);//set the initial position of car
  int a = 0;
  while (a<1) {
  arm_increase_height();
  gripper_release();// first release gripper for further action
  passive_wait(3.0);
  arm_increase_height();//move to food position and detect the food
  passive_wait(2.0);
  gripper_grip();//grip
  passive_wait(2.0);
  arm_decrease_height();//rise up to avoid collision
  passive_wait(3.0);
  arm_increase_orientation();
  arm_increase_orientation();
  arm_increase_orientation();
  arm_increase_orientation();//turn the arm
  passive_wait(3.0);
  arm_increase_height();
  passive_wait(3.0);
  gripper_release();
  passive_wait(2.0);
  arm_reset();//reset and end
  gripper_grip();
  a=2;// break from the if
  }
  wb_robot_cleanup();

  return 0;
}
```

Figure19 – Code for main function

8.2.2 ANALYSIS AND DISCUSSION

NAME: YIJUN ZHAO

My task is to detect the yellow box and control the car to move to the pond and cooperate with the mechanical Arm. I will show my ideas accompanied with the code.

8.2.2.1 CODE ANALYSIS

I first use the same function as task 1 in the main loop because there is also a straight line in the former part of task 2. And break after each loop to judge the condition again.

```
for (;;) {  // Main Loop
  // Run one simulation step
  wb_robot_step(TIME_STEP);

  for (i = 0; i < NB_GROUND_SENS; i++)
    gs_value[i] = wb_distance_sensor_get_value(gs[i]);

  // Speed initialization
   speed[LEFT]=0 ;
   speed[RIGHT]=0;

  // *** START OF SUBSUMPTION ARCHITECTURE ***

  // LFM - Line Following Module
  LineFollowingModule();

  speed[LEFT] = lfm_speed[LEFT];
  speed[RIGHT] = lfm_speed[RIGHT];


  // *** END OF SUBSUMPTION ARCHITECTURE ***


  // Set wheel speeds

  wb_motor_set_velocity(wheels[0], 0.00628 *speed[LEFT]);
  wb_motor_set_velocity(wheels[1], 0.00628 *speed[RIGHT]);
  wb_motor_set_velocity(wheels[2], 0.00628 *speed[LEFT]);
  wb_motor_set_velocity(wheels[3], 0.00628 *speed[RIGHT]);
  break;

}
}
```

Figure20 – Code for Line Follow

Then followed by the ***get_camera*** function to get use of camera and get the real time image and use the get green\blue\red function to get the RGB value of the image in camera. The RGB of yellow is (255,255,0),but there may be other color in the image to disturb the judgement ,so i use the condition of R>4G,R>4B.

```
 camera = wb_robot_get_device("camera");
 wb_camera_enable(camera, TIME_STEP);
 width = wb_camera_get_width(camera);
 height = wb_camera_get_height(camera);

for (i = width / 3; i < 2 * width / 3; i++) {
  for (j = height / 2; j < 3 * height / 4; j++) {
    red += wb_camera_image_get_red(image, width, i, j);
    blue += wb_camera_image_get_blue(image, width, i, j);
    green += wb_camera_image_get_green(image, width, i, j);
  }
}
//if (red>200&&green>200&&blue<50){
if ((red > 4 * blue) && (green > 4 * blue)){
i=0;
```

Figure21 – Code for Color Detection

When it detects the yellow box, it will end the line follow function and operate the ***turn_right*** function in the if loop. I use the wait function to control the time of each step. It need to point out that in the end of the code I let the car just move forward rather than operate the line follow function again because I will make the task easier and can also achieve the target .After all I the break function again to end task-2 and link to task-3.

```
wb_robot_step(TIME_STEP);
 speed[LEFT]=0 ;
 speed[RIGHT]=0;
 TurningRightModule();
 speed[LEFT] = lfm_speed[LEFT];
 speed[RIGHT] = lfm_speed[RIGHT];
 wb_motor_set_velocity(wheels[0], 0.00628 *speed[LEFT]);
 wb_motor_set_velocity(wheels[1], 0.00628 *speed[RIGHT]);
 wb_motor_set_velocity(wheels[2], 0.00628 *speed[LEFT]);
 wb_motor_set_velocity(wheels[3], 0.00628 *speed[RIGHT]);
printf("i find a yellow box\n", red,green,blue);
    passive_wait(0.5);

 wb_motor_set_velocity(wheels[0], 10);
 wb_motor_set_velocity(wheels[1], 10);
 wb_motor_set_velocity(wheels[2], 10);
 wb_motor_set_velocity(wheels[3], 10);
 passive_wait(1.8);

wb_motor_set_velocity(wheels[0], 0);
 wb_motor_set_velocity(wheels[1], 0);
 wb_motor_set_velocity(wheels[2], 0);
 wb_motor_set_velocity(wheels[3], 0);
    passive_wait(22);
    wb_motor_set_velocity(wheels[0], -10);
 wb_motor_set_velocity(wheels[1], -10);
 wb_motor_set_velocity(wheels[2], -10);
 wb_motor_set_velocity(wheels[3], -10),
 passive_wait(1.60);
 TurningRightModule();
 speed[LEFT] = lfm_speed[RIGHT];
 speed[RIGHT] = lfm_speed[LEFT];
 wb_motor_set_velocity(wheels[0], 0.00628 *speed[LEFT]);
 wb_motor_set_velocity(wheels[1], 0.00628 *speed[RIGHT]);
 wb_motor_set_velocity(wheels[2], 0.00628 *speed[LEFT]);
 wb_motor_set_velocity(wheels[3], 0.00628 *speed[RIGHT]);
 passive_wait(0.481);
  wb_motor_set_velocity(wheels[0], 15);
 wb_motor_set_velocity(wheels[1], 15);
 wb_motor_set_velocity(wheels[2], 15);
 wb_motor_set_velocity(wheels[3], 15);
    passive_wait(1);
```

Figure22 – Code for Turning and Back

8.2.2.2 TECHNICAL PROBLEM

At first, I do not know how to get the RGB value of the image so I search the ***WEBOT*** guide tour and know the get ren green blue function

It spent me plenty of time to figure how to build the loop, and finally I figure out the break function and let the judgement operated every single time to judge which loop to operate.

The next problem is how to control the car to turn tight 90 degree, I do not know compass at that time so I can just use wait function to control the time of each step and try so many times to find the correct time for right angle. But when I finish all the jobs, I know that I can use compass to turn accurate 90-degree angle

The final problem is that I need to let the car go closed to the bridge so that task3 can be successfully operated. the first attempt is to let the line to be closer to the bridge but I find that wheels may crash to the bridge. So I finally make a fine adjustment of the angle of the turning and achieve success.

8.3.1 ANALYSIS AND DISCUSSION

NAME: JINGYUAN FENG

My job was to make the car detect the bridge and cross it safely. So that this part is going to be separated into two parts.

8.3.1.1 BRIDGE DETECTING

The trolley relies mainly on distance sensors to detect the position of the bridge.

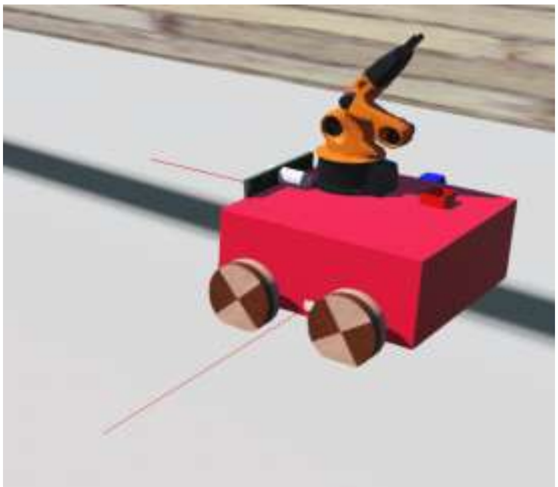


Figure23 – Sensors Lines

We can see from the figure that there is a thin red line on the left side of the car, which represents the detection distance of the ***distance sensor*** installed on the car. When there is an obstacle in contact with it, the ***distance sensor*** will be activated and the car will start to turn through the code. In addition, it is not difficult to notice that the ***distance sensor*** on the left side of the car is installed in a relatively low position. This is because I found in the test that the bridge model is relatively low, and if the sensor position is higher, it will not be easy to detect the bridge. Therefore, I adjusted the position of the sensor.

```
WbDeviceTag ds[1];
char ds_names[1][10] = {"ds_left"};
for (i = 0; i < 1; i++) {
  ds[i] = wb_robot_get_device(ds_names[i]);
  wb_distance_sensor_enable(ds[i], TIME_STEP);
}
```

Figure24 – Connection Code

The code above is the connection between the ***distance sensor*** and the entire car system. And here we notice that I'm using the ***wb_distance_sensor_enable*** function, This function is used to activate the ***distance sensor***, so we can only call it when it is activated, Also the larger ***TIME_STEP*** is, the faster the numerical refresh of the sensor will be.

8.3.1.2 TURNING OF THE CAR

Getting the car to turn at the right Angle is crucial to getting the car safely across the bridge.

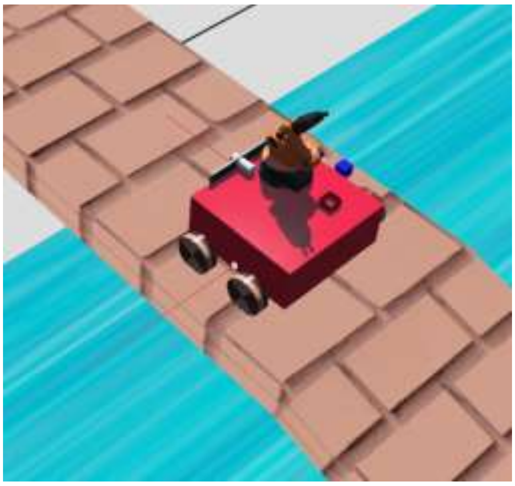


Figure25 – Bridge Crossing and Sensor Lines

```
if (avoid_obstacle_counter > 0) {
  avoid_obstacle_counter--;
  left_speed = -2.1;
  right_speed = 2.1;
  if(avoid_obstacle_counter ==0)
    break;

} else { // read sensors
  double ds_values[1];
  for (i = 0; i < 1; i++)
    ds_values[i] = wb_distance_sensor_get_value(ds[i]);
  if (ds_values[0] <999.0 )
    avoid_obstacle_counter = 57;
}
```

Figure26 – Avoid Obstacle Code

We spent a lot of time making adjustments so that the car could rotate at the right Angle. Here we control the car's final turning Angle by controlling the car's rotation time and the car's wheels' speed, the ***avoid_obstacle_counter*** is like a countdown timer set for 57 units of time and here the car's left wheels' speed was set to -2.1 while the car's right wheels' speed was set to 2.1 at the same time to make car can rotate by 90 degree.

8.3.2 TREES AVOIDANCE

NAME: ZHUOZHAO LIU

8.3.2.1 TREES AVOIDANCE

Originally, I built a model with two distance sensors on the left and right front side of the car, so it can detect both sides in front and make a turn if the obstacle is found.

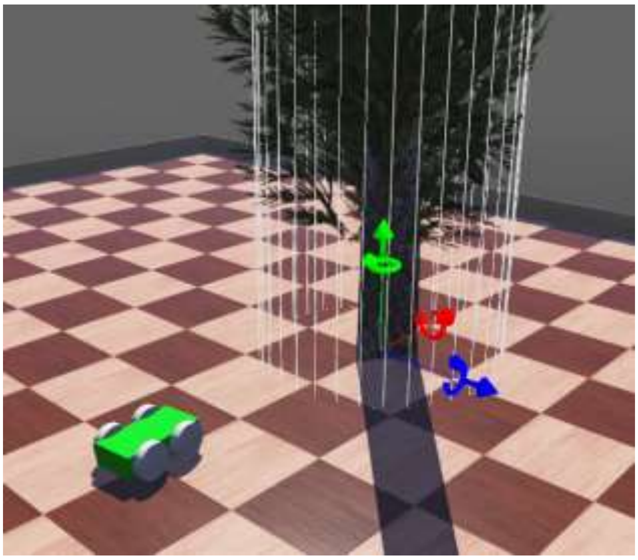


Figure27 – Testing Model for Task-4

The main part of codes are as follows,

```
#define TIME_STEP 64
using namespace webots;

int main(int argc, char **argv) {
  Robot *robot = new Robot();
  DistanceSensor *ds[2];
  char dsNames[2][10] = {"ds_right", "ds_left"};
  for (int i = 0; i < 2; i++) {
    ds[i] = robot->getDistanceSensor(dsNames[i]);
    ds[i]->enable(TIME_STEP);
  }
  Motor *wheels[4];
  char wheels_names[4][8] = {"wheel1", "wheel2", "wheel3", "wheel4"};
  for (int i = 0; i < 4; i++) {
    wheels[i] = robot->getMotor(wheels_names[i]);
    wheels[i]->setPosition(INFINITY);
    wheels[i]->setVelocity(0.0);
  }
  int avoidObstacleCounter_1 = 0;
  int avoidObstacleCounter_2 = 0;
  while (robot->step(TIME_STEP) != -1) {
    double leftSpeed = 5.0;
    double rightSpeed = 5.0;
    if (avoidObstacleCounter_1 > 0) {
      avoidObstacleCounter_1--;
      leftSpeed = 2.1;
      rightSpeed = -2.1;
    if (avoidObstacleCounter_2 > 0) {
      avoidObstacleCounter_2--;
      leftSpeed = -2.1;
      rightSpeed =2.1;
    } else { // read sensors
      for (int i = 0; i < 2; i++) {
        if (ds[i]->getValue() < 950.0)
          avoidObstacleCounter = 57;
      }
    }
```

Figure28 – Main Codes for Task-4

After defining wheels, I use ***avoid_Obstacle_Counter*** to let the car make a turn. After adjusting the parameters, I found the value of 57 is suitable for the avoidance of trees and make a turn to finish the following tasks. By changing the positive or negative speed of the car, its turning position is different. A pointer function is used in this code and use the function ***di[i]->getvalue()*** to get the return value to decide the time to make a turn. If it's less than 950+, the car will stop and make a turn. The trees' model is a cylindrical model, so the car just needs to find the mode and do the above steps and make a turn.

8.3.2.2 COMBINATION WITH OTHER TASKS

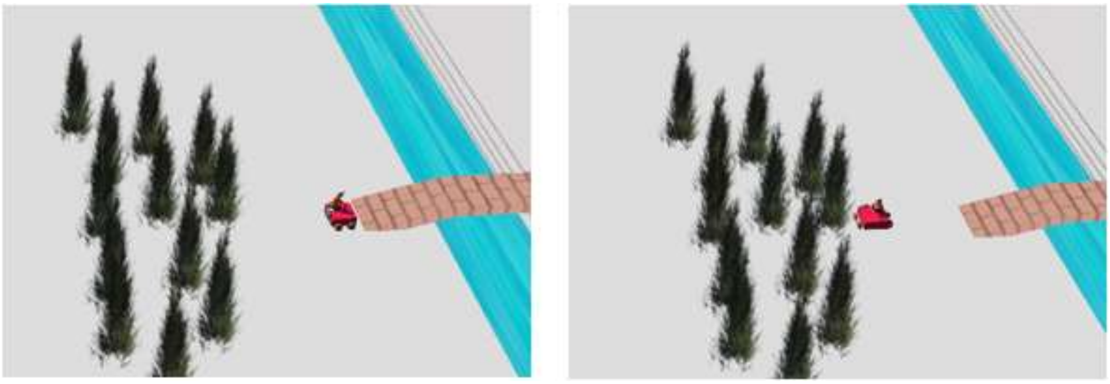


Figure29 – Situation After Crossing Bridge

In this combination, I use just one distance sensor to make a change to fit in with other parts, and didn't use a pointer function. It can also work well and make a suitable turn. The main codes modified are as follows:

```
while (wb_robot_step(TIME_STEP) != -1) {
  double left_speed = 5;
  double right_speed = 5;
  if (avoid_obstacle_counter1 > 0&& avoid_obstacle_cou
   {
    avoid_obstacle_counter1--;
    left_speed = -2.1;
    right_speed = 2.1;
    if(avoid_obstacle_counter1==0)
    {avoid_obstacle_counter4=1; }}
  else if ((avoid_obstacle_counter1 > 0)&&(avoid_obsta
    avoid_obstacle_counter1--;
    left_speed = -2.1;
    right_speed = 2.1;
    if(avoid_obstacle_counter1 ==0)
    break;
  }

 else if (avoid_obstacle_counter2 > 0) {
    avoid_obstacle_counter2--;
    left_speed = 2.1;
    right_speed = -2.1;
    avoid_obstacle_counter3=1;
    if(avoid_obstacle_counter2==0)
    {left_speed = 5;
    right_speed = 5;
    wb_motor_set_velocity(wheels[0], left_speed);
    wb_motor_set_velocity(wheels[1], right_speed);
    wb_motor_set_velocity(wheels[2], left_speed);
    wb_motor_set_velocity(wheels[3], right_speed);
    passive_wait(6);
    avoid_obstacle_counter3=0;
    }
```

```
     else{ // read sensors
      double ds_values1[1];
      for (i = 0; i < 1; i++)
        ds_values1[i] = wb_distance_sensor_get_value(ds[
      if (ds_values1[0] <999.0 )
        {passive_wait(1);
        avoid_obstacle_counter1 = 107;}
      double ds_values2[1];
      for (i = 0; i < 1; i++)
        ds_values2[i] = wb_distance_sensor_get_value(ds_
      if (ds_values2[0] <999.0 )
        avoid_obstacle_counter2 = 117 ;
    }
    wb_motor_set_velocity(wheels[0], left_speed);
    wb_motor_set_velocity(wheels[1], right_speed);
    wb_motor_set_velocity(wheels[2], left_speed);
    wb_motor_set_velocity(wheels[3], right_speed);
  }

double left_speed = 5;//
double right_speed = 5;
wb_motor_set_velocity(wheels[0], left_speed);
wb_motor_set_velocity(wheels[1], right_speed);
wb_motor_set_velocity(wheels[2], left_speed);
wb_motor_set_velocity(wheels[3], right_speed);
//task5
```

Figure30 – Solutions to The Collision

It defined some counters instead of one. In order to turn by using ***counter--,*** we set its initial value to 0, and ***avoid_obstacle_counter*** to some certain values as above. After getting the return value to less than 999, the car makes a turn to connect with the next step.

8.4.1 ANALYSIS AND DISCUSSION

NAME: YUXIAO LUO

The first technical problem which we faced is how to detect the arch. I have communicated it with Xingyue Wu, who is my partner in task-4, finally, compared with camera, we used the distance sensor. The benefit is that it can detect the bridge, the trees and the arch, whatever is shapes and colors. And it easier to control than the camera. The second problem is how to combine task-3 and task-4, and it will turn twice in the tree. I solved this problem by myself, through optimizing the codes which I mentioned in Technical content.

8.4.2 ANALYSIS AND DISCUSSION

NAME: XINGYUE WU

8.4.2.1 SELECT AND ADD SENSORS:

We select the distance sensor to detect the arch.

We plan to add a distance sensor on the left side of the car. Set its graphical and physical shape to a cube (not transformed) having the edge of 0.02[m]. Set their color to white. Set their name: ***ds_left***.

To add the distance sensors to the robot, we add a DistanceSensor node as direct children of the Robot node. Due to the distance sensor acquires its data along the x-axis, we rotate the distance sensors in order to point their x-axis outside the robot. (see the figure).

Figure31 – Sensors and the Lines

8.4.2.2 DESIGN AND PROBLEMS SOLUTIONS

Our plan is that when the sensor on the left recognizes the left side of the arch bridge, the car will turn left 90 degrees and pass the arch bridge.

This creates two problems: The recognition range of the sensor and how to achieve an accurate rotation of 90 degrees.

We found that the range that the sensor should be set based on the linear distance between the car and the arch bridge. Before identifying the sensor, the cart travels straight, so this distance is the same every time. Since the ray from the left distance sensor is perpendicular to the body, the recognizable range of the sensor should be greater than or equal to the linear distance from the arch bridge when the car reaches the arch bridge.

We changed the code so that the car can rotate 90 degrees. After that, the code 'break' can help the car to stop turning and go through the arch.

Here is the code we test to realize 90-degree turning:

```c
#include <webots/distance_sensor.h>
#include <webots/motor.h>
#include <webots/robot.h>

#define TIME_STEP 64

int main(int argc, char **argv) {
  wb_robot_init();
  int i;
  bool avoid_obstacle_counter = 0;

  WbDeviceTag ds[1];
  char ds_names[1][10] = {"ds_left"};
  for (i = 0; i < 1; i++) {
    ds[i] = wb_robot_get_device(ds_names[i]);
    wb_distance_sensor_enable(ds[i], TIME_STEP);
  }
  WbDeviceTag wheels[4];
  char wheels_names[4][8] = {"wheel1", "wheel2", "wheel3", "wheel4"};
  for (i = 0; i < 4; i++) {
    wheels[i] = wb_robot_get_device(wheels_names[i]);
    wb_motor_set_position(wheels[i], INFINITY);
  }

 while (wb_robot_step(TIME_STEP) != -1) {
   double left_speed = 1.0;
   double right_speed = 1.0;
   if (avoid_obstacle_counter > 0) {
     avoid_obstacle_counter--;
     left_speed = -1.0;
     right_speed = 1.0;
     if(avoid_obstacle_counter ==0)
       break;

   } else { // read sensors
     double ds_values[1];
     for (i = 0; i < 1; i++)
       ds_values[i] = wb_distance_sensor_get_value(ds[i]);
     if (ds_values[0] <999.0 )
       avoid_obstacle_counter = 57;
   }
   wb_motor_set_velocity(wheels[0], left_speed);
   wb_motor_set_velocity(wheels[1], right_speed);
   wb_motor_set_velocity(wheels[2], left_speed);
   wb_motor_set_velocity(wheels[3], right_speed);
 }
 double right_speed = 1.0;
 double left_speed = 1.0;
 wb_motor_set_velocity(wheels[0], left_speed);
 wb_motor_set_velocity(wheels[1], right_speed);
 wb_motor_set_velocity(wheels[2], left_speed);
 wb_motor_set_velocity(wheels[3], right_speed);

  wb_robot_cleanup();
  return 0;  // EXIT_SUCCESS
}
```

Figure32 – Code to Realize 90-degree Turning

When the car detects obstacles, the value of ***avoid_obstacle_counter*** is equal to 57.

```c
if (ds_values[0] <999.0 )
  avoid_obstacle_counter = 57;
```

Figure33 – Avoid Obstacle Counter

When the value of ***avoid_bostacle_counter*** is more than 0, the car begins to turn due to the speed difference of the left and right wheels and at the same time. And when the value is 0, the car turns nearly 90 degrees, and then ‘break’. (57 is set by testing plenty of times).

```c
if (avoid_obstacle_counter > 0) {
  avoid_obstacle_counter--;
  left_speed = -1.0;
  right_speed = 1.0;
  if(avoid_obstacle_counter ==0)
    break;
```

Figure34 – Turning Order

After our discussion, Yuxiao Luo gives strong technical support in programming. And the solutions are optimized in the final results.

8.5.1 ANALYSIS AND DISCUSSION

NAME: YIMIN TIAN

8.5.1.1 DESIGN OF ROUTE MAPS

The objective here is to paint the colored lines for line following according to the patio documents given on Moodle.

Lines:

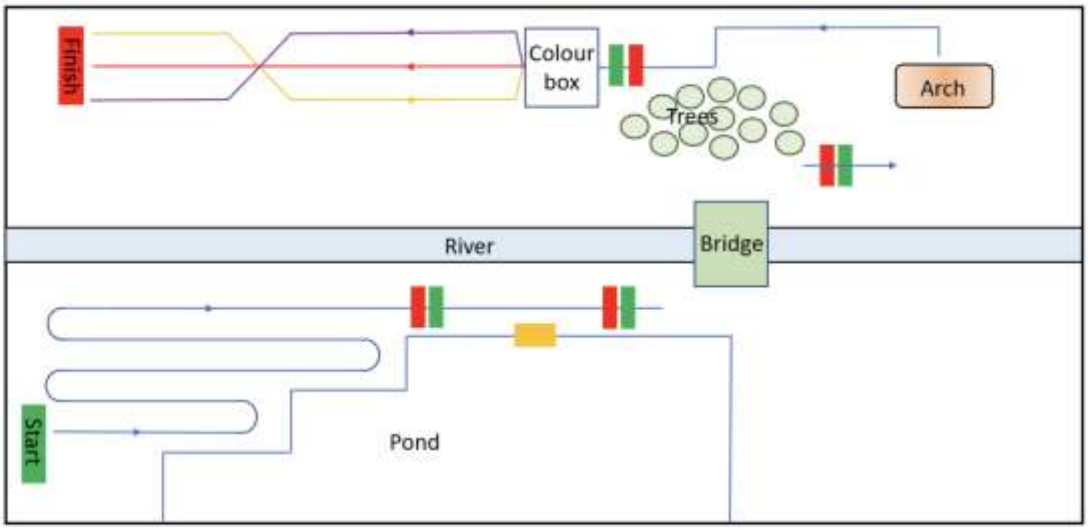


Figure35 – Patio Picture from Designtaskan-overview.ppt

In task-5, lines between arch and last finish line need to be done. ***Microsoft Paint 3D***available on Windows 10, is introduced in line painting as its canvas is resized easily to cater requirement from ***WEBOT*** and adequate 2D Shape options is convent at designing curve and changing color.

Figure36 – 2D Tools

Figure37 – Resize Options

The final route map of Task-5 is shown below:

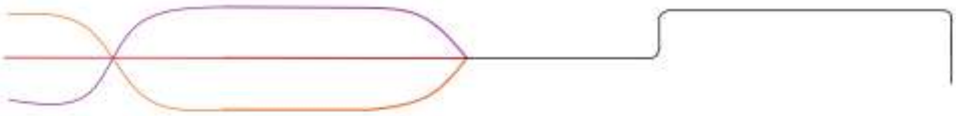


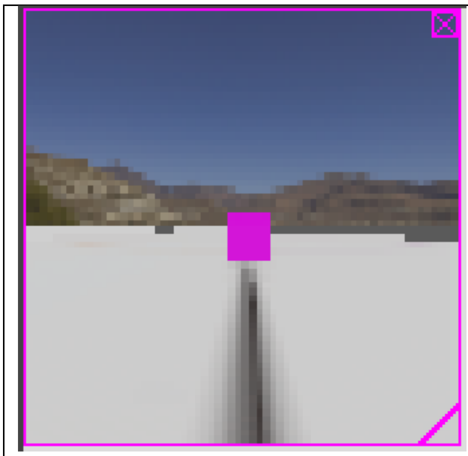
Figure38 – The Final Route Map of Task-5

8.5.1.2 RECOGNIZE COLOR BOX

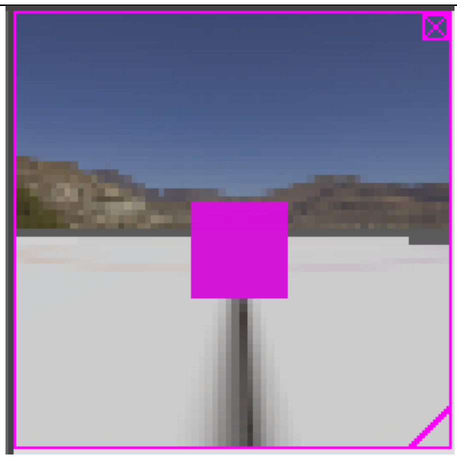
Camera node brings a series of API to handle image analysis.

What camera functions Task 5 has used is:

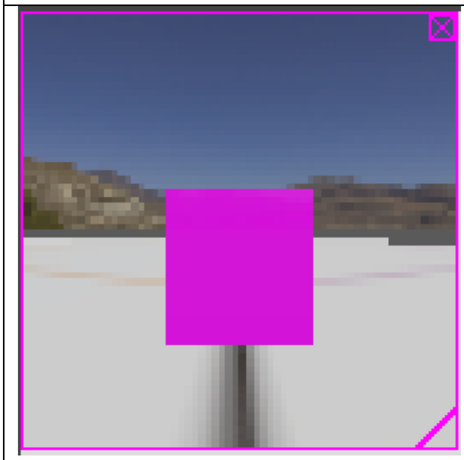
```
Camera {
SFInt32 width      64      # [0, inf)
SFInt32 height     64      # [0, inf)
}
```



Frame 1



Frame 2



Frame 3

Figure39 – Statistics for Each Frame Captured by the Camera

8.5.1.3 HOW TO RECOGNISE RED, YELLOW, PURPLE BOX?

One part of Task-5 is to choose one of three colored line to do line following according to the colored box.

When car, in the line following status, reaches colored box, central process unit is requested to analyze frame captured by camera at this moment and to recognize color for next step.

But how to determine present frame do contain a colored box?

One way is calculating RGB pixels and find RGB relation for each colored boxes.

The use of HSV as a color space for the visual system mitigates the effects of site lighting changes on the visual system.

For strategy one, we use ***wb_camera_get_width*** ***wb_camera_get_height*** to get present frame size. With the help of ***wb_camera_image_get_red***

wb_camera_image_get_green wb_camera_image_get_blue, pixels of red, green and blue can be directly count out.

1. For frame 1

Red	Green	Blue
64321	58271	64321

Table4 – RGB for Frame 1

2. For frame 2

Red	Green	Blue
65443	48576	65702

Table5 – RGB for Frame 2

3. For frame 3

Red	Green	Blue
69145	21183	70329

Table6 – RGB for Frame 3

Consider the multiplicative relationship of these three colors, it is easy to say when both red and blue pixels is three times than green one, camera find a purple box. Which write in C language is:

Purple

if ((red > 3 * green) && (blue > 3 * green))

With same procedures, other two colored-boxes can be recognized by:

Red

if ((red > 3 * green) && (red > 3 * blue))

Yellow

if ((red > 3 * blue) && (green > 3 * blue))

8.5.1.4 SUMMARY

As original RGB model can handle these three colored boxes very well, more complex HSV color space will not be applied in Task-5. Task-5 will use ***wb_camera_get_width***

wb_camera_get_height to get present frame size and use ***wb_camera_image_get_red*** ***wb_camera_image_get_green*** ***wb_camera_image_get_blue***, pixels of red, green and blue to count each color pixels out. With help of the recognition conditions, camera module has ability to distinguish whether present frame contains a colored box or not.

8.5.1.4 HOW TO FOLLOW THE COLORED LINE?

Line following module needs to be improved as line in task 5 is not just black line in task 1 anymore. To be more specific, infra-red sensors, used in task 1, cannot really distinguish the difference between yellow line and white ground, as yellow in spectrum is close to red.

If we just directly use task-1 line following module in task-5 yellow line,

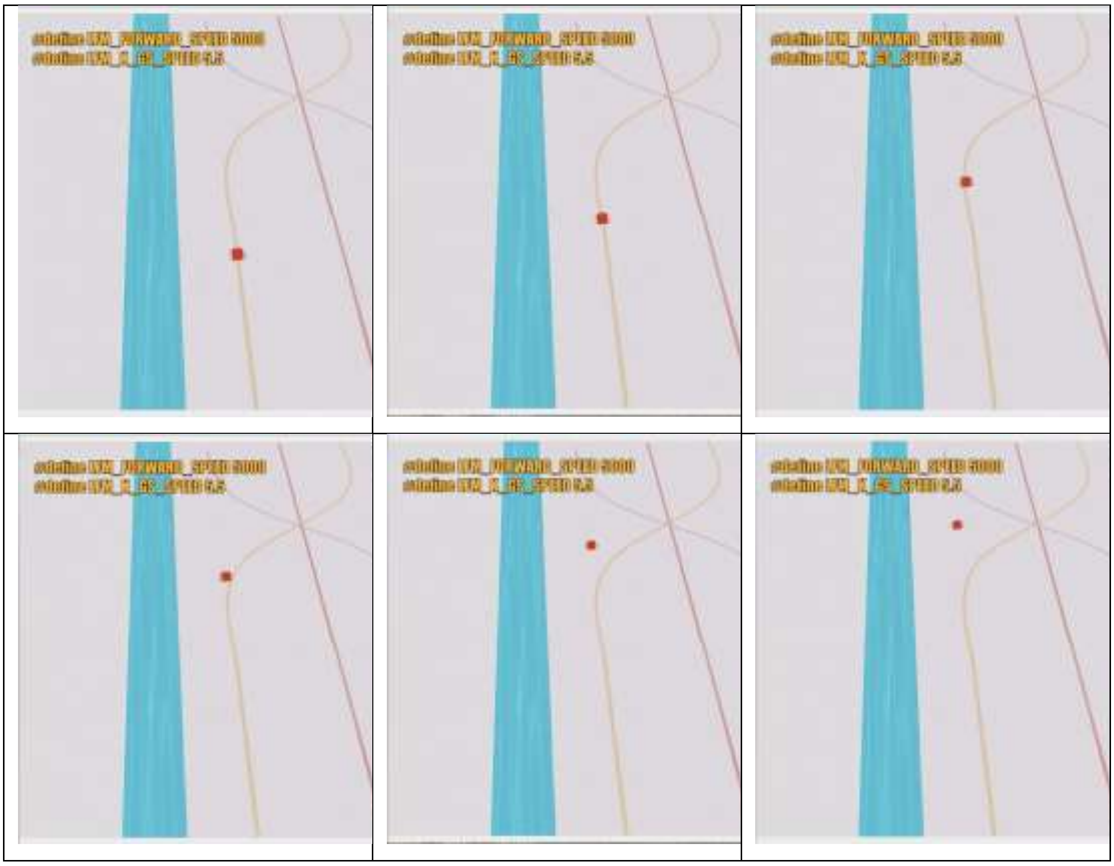


Figure40 – Test Result

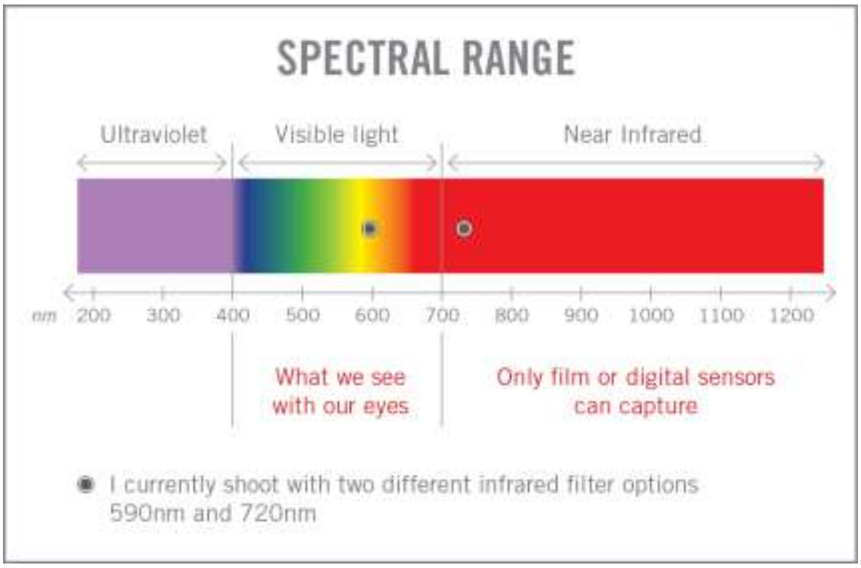


Figure41 – Spectral Range

Stay with task 1 solution, but use more sensitive parameters to enable sensor to notice minor difference between colored line and white background.

```
void LineFollowingModule(void) {  int DeltaS = gs_value[GS_RIGHT] -
gs_value[GS_LEFT];  lfm_speed[LEFT] = LFM_FORWARD_SPEED -
LFM_K_GS_SPEED *DeltaS;  lfm_speed[RIGHT] = LFM_FORWARD_SPEED +
LFM_K_GS_SPEED *DeltaS; }
```

Use visual line follow solution, such as using camera to process image and to obtain an abstract mathematical geometric model.

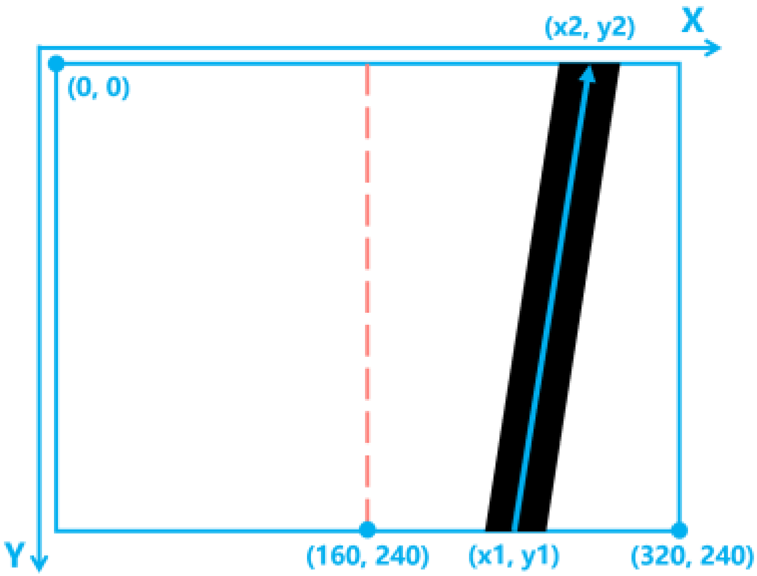


Figure42 – Plan 2 Visual Line Following Solution

For plan 2, use camera to do line following is a complex job, as image have to get pre-processing to become a measurable and countable model. One thing about simulation performance of ***WEBOT*** may not be enough to process information in time. Thus, we take plan 1 as first test turn.

TEST PART

For plan 1, These are the parameters we need to adjust.

#define LFM_FORWARD_SPEED 5000 #define LFM_K_GS_SPEED 5.5

LFM_FORWARD_SPEED regulars initial speed, ***LFM_K_GS_SPEED*** regulars level of adjustment

The objective of this part is to let car follow the yellow lines by adjusting ***LFM_FORWARD_SPEED*** and ***LFM_K_GS_SPEED*** parameters.

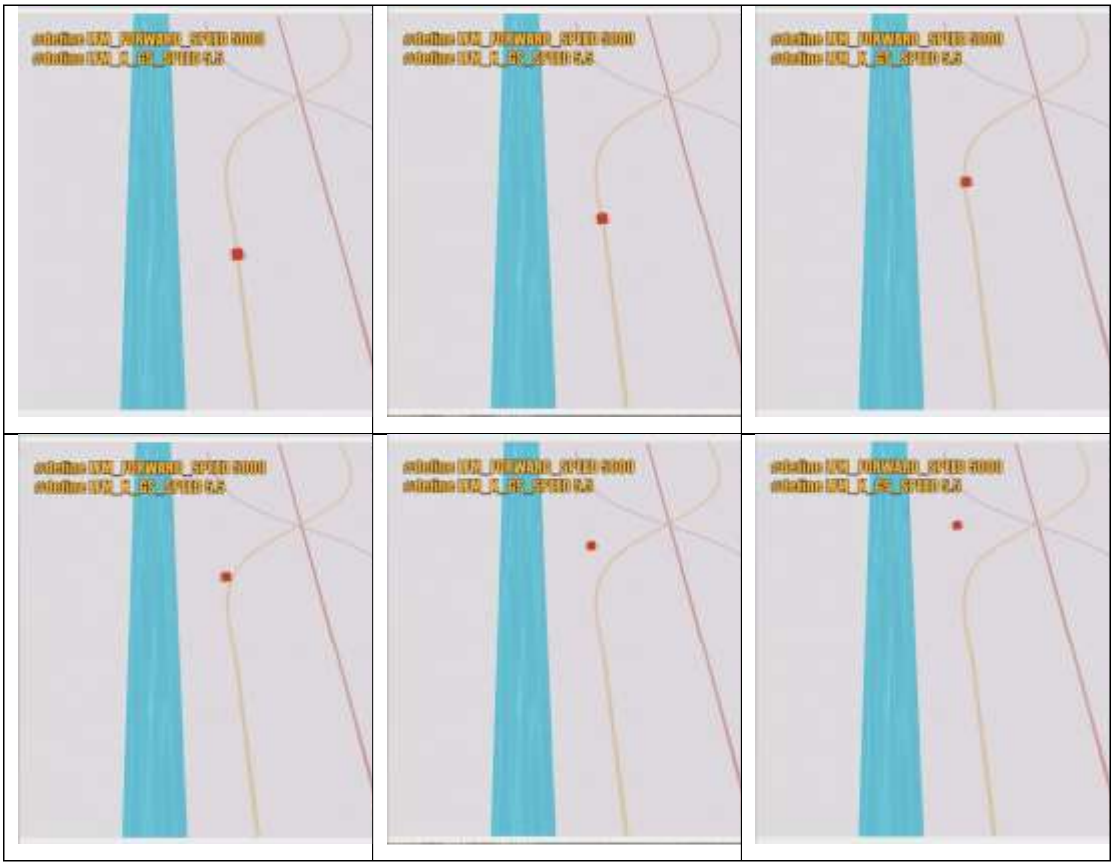


Figure43 – Test 1

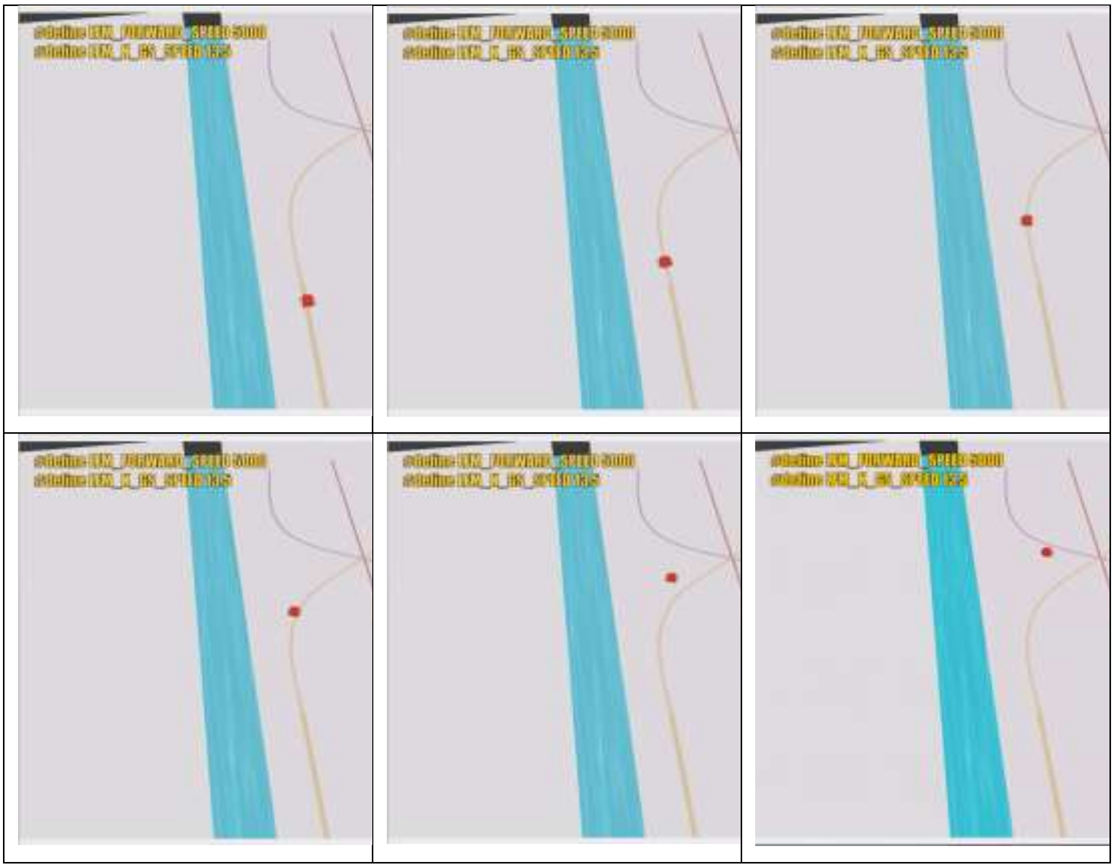


Figure44 – Test 2

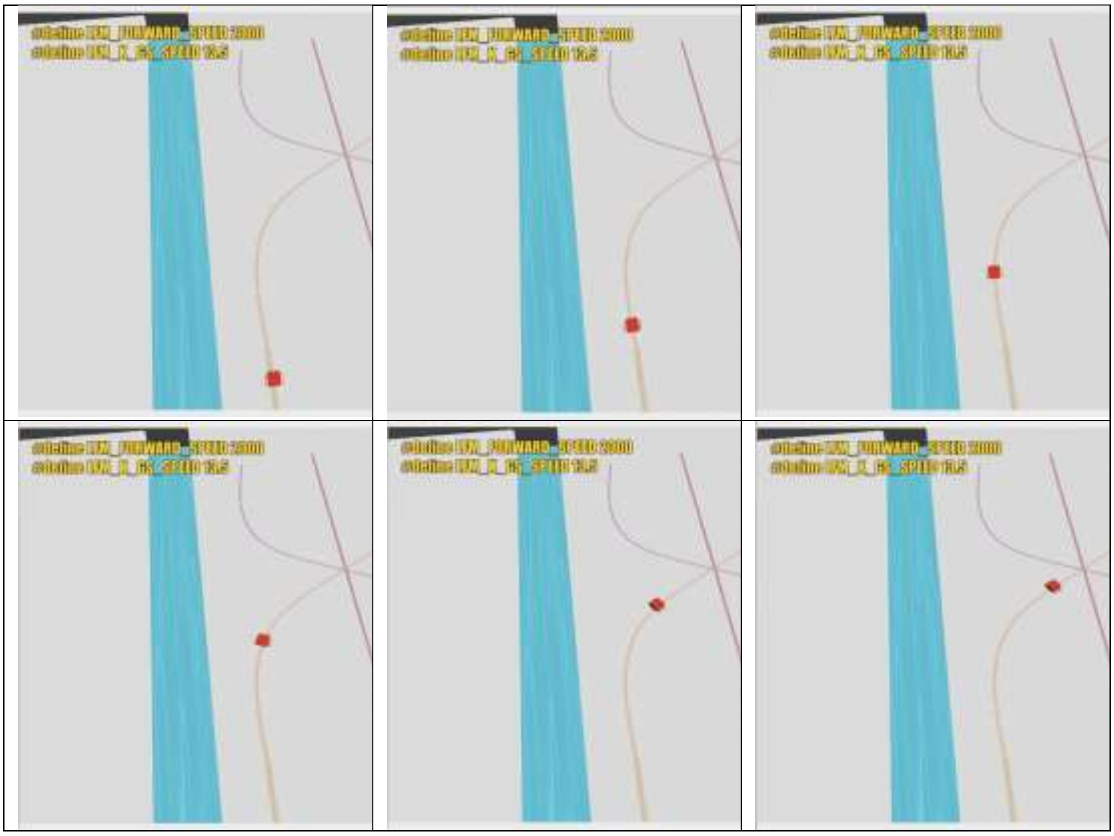


Figure45 – Test 3

With several tests, infra-red sensor can continuously detect difference between the colored line and the background, and the final parameters setting is as:

#define LFM_FORWARD_SPEED 2000 #define LFM_K_GS_SPEED 13.5

Because 'plan 1' has already handled colored line following very well, we did not take more tests on 'plan 2'.

8.5.2 ANALYSIS AND DISCUSSION

NAME: WENTAO DU

8.5.2.1 ANALYSIS

The first step is to add camera and add "***transform***" in the children of camera. (The white cylinder is ***transform*** and the blue electromagnetic plate is ***distance sensor***). Then, our car can receive information from the patio.

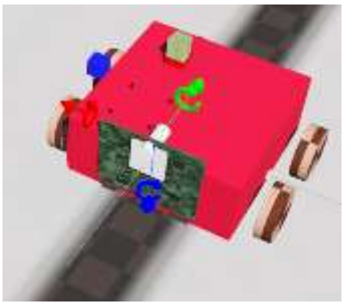


Figure46 – Conceptual Graph

From the readily available controller in ***WBOT***, the codes taught me how to use RGB to judge the color of an item and an algorithm controlling line following.

```
int i, j;
int red, blue, green;
const unsigned char *image = wb_camera_get_image(camera);
red = 0;
green = 0;
blue = 0;
for (i = width / 3; i < 2 * width / 3; i++) {
  for (j = height / 2; j < 3 * height / 4; j++) {
    red += wb_camera_image_get_red(image, width, i, j);
    blue += wb_camera_image_get_blue(image, width, i, j);
    green += wb_camera_image_get_green(image, width, i, j);
  }
}
```

Figure47 – Color Detection Code

It int RGB and i & j which is used in a for loop with the width and height we set before to control the value of RGB. It also const image which get data from the camera. Then we can get the value of RGB comes from image, width, i and j. The color of an item can be defined by different value of RGB and we can check it on the internet. For example, ***（red>>3*green）&&(blue>>3*green)*** means the box is purple.

```
void LineFollowingModule(void) {
  int DeltaS = gs_value[GS_RIGHT] - gs_value[GS_LEFT];

  lfm_speed[LEFT] = LFM_FORWARD_SPEED - LFM_K_GS_SPEED *DeltaS;
  lfm_speed[RIGHT] = LFM_FORWARD_SPEED + LFM_K_GS_SPEED *DeltaS;
}
```

Figure48 – Line Follow Mode

This module realizes line following by giving different speed to left or right wheels. The next step is to write a code issuing command to the car after it has known the color of the box. The car has to go on different road, so it may turn to some direction at first. Just like the module of line following.

Later we can call our module easily.

8.5.2.2 SOLUTIONS

I have two guesses for solving problems. Both solutions used if sentences:

Without pre-set counter

If the car sees a yellow box it will turn right, else if the car has rotated to a certain angle at which it cannot see the color box any more it will execute the function line following. In this way we can realize our target in a while loop with many if sentences in it. However, we give this idea up, because the color of the road may influence the camera when car drive on the road, which means the car cannot drive stably.

(With pre-set counter to control how much different operations last)

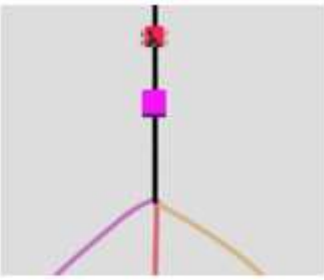
The other solution is we used a timer set previously to control how much time the turning operation last, so the car will turn to the road with color the same as box. In this way our if sentences only execute one time.

```
if ((red > 3 * green) && (red > 3 * blue)){
  current_blob = 1;//purple
  printf("Looks like I found a %d blob.\n", current_blob);


}
```

Figure49 – Line Chosen Mode

This is the one condition of the if sentences to recognize color and what the car will do after passing through the box is included in the if sentence (write codes at the white space remained in if sentence).



```
int a=0;
while ((wb_robot_step(TIME_STEP) != -1) & (a<= 130)){
speed[LEFT]= GEN_SPEED;
speed[RIGHT]= GEN_SPEED;
wb_motor_set_velocity(wheels[0], 0.01628 *speed[LEFT]);
wb_motor_set_velocity(wheels[1], 0.01628 *speed[RIGHT]);
wb_motor_set_velocity(wheels[2], 0.01628 *speed[LEFT]);
wb_motor_set_velocity(wheels[3], 0.01628 *speed[RIGHT]);
a = a+1;
}
```

Figure50 – Selection of Path

As we can see when the car recognize the color, it will keep going forward and the time is controlled by the timer, which means when time is over, the car arrives at the fork in the road and it will execute the next step ”turning to the road in purple”. Also, the turning operation execute for a pre-defined time controlled a timer, so it rotates to a suitable angle and when the time is over it continues executing line following module. The codes are very similar as before, but change the speed of wheels into turning module and line following module.

8.5.2.3 DISCUSSION:

Actually, solution2 do ensure the stability of the whole process, because the car will always drive as we set before, it will keep turning right, turning left and going forward for a invariant time, which I mean is it will never make mistake when running on the ***WEBOT***. However, the real road is influenced by many factors, it keeps changing all the time. We can decrease the width and height of the camera to avoid it misleading by the color of road.

8.5.3 ANALYSIS AND DISCUSSION

NAME: MIANZHE WU

Firstly, I need to adjust the direction of car after it detects the color, so I need to change the speed of four steering wheels to generate speed difference so the car can turn its direction, here is its code

```
a=0;
while (wb_robot_step(TIME_STEP) != -1 & a<= 80){
speed[LEFT]= GEN_SPEED;
speed[RIGHT]= NO_SPEED;
wb_motor_set_velocity(wheels[0], 0.00628 *speed[LEFT]);
wb_motor_set_velocity(wheels[1], 0.00628 *speed[RIGHT]);
wb_motor_set_velocity(wheels[2], 0.00628 *speed[LEFT]);
wb_motor_set_velocity(wheels[3], 0.00628 *speed[RIGHT]);
a = a+1;
}
```

Figure51 – Code for Setting Velocity

In this while loop, first I set the value of parameter a to be 0, then in this loop I set the speed of two wheels which go forward to be ***GEN_SPEED*** and ***NO_SPEED***, then the following four lines represent the speed of four steering wheels, in this way we can accomplish the task of changing directions.

Next, I need to let the car follow the line of color it detects. Since this method is similar with that in task1, so I ask my teammates for help. Line following is mainly based on infrared sensor which can detect the difference of color between the lines and ground then the car can follow lines. Here is the code.

```
for (;;) {  // Main loop
  // Run one simulation step
  wb_robot_step(TIME_STEP);
  for (i = 0; i < NB_GROUND_SENS; i++)
    gs_value[i] = wb_distance_sensor_get_value(gs[i]);

  // Speed initialization
  speed[LEFT]=0 ;
  speed[RIGHT]=0;
  LineFollowingModule();
  speed[LEFT] = lfm_speed[LEFT];
  speed[RIGHT] = lfm_speed[RIGHT];
  wb_motor_set_velocity(wheels[0], 0.00628 *speed[LEFT]);
  wb_motor_set_velocity(wheels[1], 0.00628 *speed[RIGHT]);
  wb_motor_set_velocity(wheels[2], 0.00628 *speed[LEFT]);
  wb_motor_set_velocity(wheels[3], 0.00628 *speed[RIGHT]);
}
```

Figure52 – Code for Selecting Path

The first part is to set the distance sensor which is used for the car to follow lines. the last line can get value from distance sensor and then guide the car. The next part is to let car follow lines and this part is similar with before. First the speed is 0 then the car accelerates, finally it can follow lines and reach finish line.

Technical problems

1. There is something wrong when the car changes its direction, after discussion I think it’s better if we can set a compass.

2. In the process of color recognition, we are asked to change the color box manually, but I wonder if we can set the color box change its color automatically.

9. System Integration, Results and Discussion

9.1.1 SYSTEM INTEGRATION OF TASK-2 AND TASK-1[12]

HARDWARE

9.1.1.1 KUKA's youBot MOBILE ROBOTIC

we move the pedestal and just use the mechanical Arm. We set it on the car and adjust the mass of the car and Arm to make sure it can successfully finish taks-1.

9.1.1.2 Food

We set a food in cube shape and build wall around it to prevent it drop from the car. Then We adjust the height and position of the wall several times in order to find the parameter that the food won't drop from the car and won't be stuck by the wall.

9.1.1.3 Camera

The camera is fixed at the right of the car because the pond is in the right side.

SOFTWARE

the arm and the car have different controller so there may not be any collision

we add a break in the "for" loop of task1 to let the judgement operate every single time. If the camera detects the yellow box, it will operate the "if" loop which is the code of task2.

```
for (i = width / 3; i < 2 * width / 3; i++) {
  for (j = height / 2; j < 3 * height / 4; j++) {
    red += wb_camera_image_get_red(image, width, i, j);
    blue += wb_camera_image_get_blue(image, width, i, j);
    green += wb_camera_image_get_green(image, width, i, j);
  }
}
//if (red>200&&green>200&&blue<50){
if ((red > 4 * blue) && (green > 4 * blue)){
i=0;
```

Figure53 – Code for main function

[12] Yijun Zhao 2357836Z 2017200604030

9.1.2 RESULTS AND DISCUSSION BY WEIXI XIONG[13]

9.1.2.1 CHOICE OF MECHANICAL ARM

Finally, ***KUKA's youBot MOBILE ROBOTIC*** is selected as our mechanical arm. The discussion is about other selections.

Discussion: Building my own mechanical arm was an optional and tested. But it is discovered that the ***boundingobject*** is hard to set as coordinate boxes are needed. So that the box which holds the fish food is hard to not easy to build.

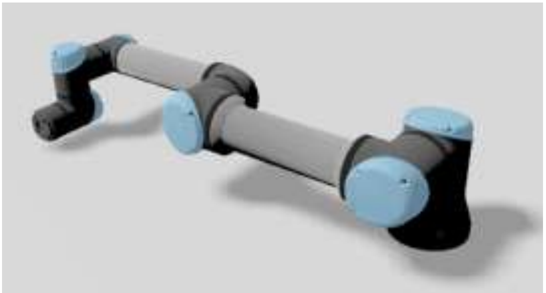


Figure54 – UR3e, 5e and 10e

UR3e, 5e and 10e is also considered. But the arm is not flexional so the arch crossing can be influenced. And that arm has too many degrees of freedom so that it is kind of waste in our project.

Figure55 – Unimation PUMA 560

Unimation PUMA 560 is another selection. Just as the one in ***KUKA's youBot MOBILE ROBOTIC***, this arm is with a pedestal but it is too high and may cause the height of the robot out of control. And also, the degree of freedom is not enough in our task as the food need to be gripped and release into the pond such that the motion rang is much wider.

[13] Weixi Xiong 2357797X 2017200603001

9.1.2.2 TRIGGER OF THE ARM

In our project, the trigger of the arm is a ***passive_wait()*** function that are provided in the default controller of ***KUKA's youBot MOBILE ROBOTIC***. The function is provided in following figure:

```
static void step() { //define function step
  if (wb_robot_step(TIME_STEP) == -1) {
    wb_robot_cleanup();
    exit(EXIT_SUCCESS);
  }
}

static void passive_wait(double sec) {
  double start_time = wb_robot_get_time();
  do {
    step();
  } while (start_time + sec > wb_robot_get_time());
}
```

Figure56 – Code for passive_wait()

The reason why it is feasible is that the simulation is under ideal condition. But if progress is needed or the practical condition is provided, then an extra camera for color detection need to be added to the car. This is because the controller of the arm is apart from that of the vehicle. The motors need a controller to be triggered when yellow is detected. Meanwhile, the parameters of the camera can not be shared through different controller. Such that the two cameras need to be exactly the same and the position difference is supposed to be very low. And the trigger condition for arm and motors could be definitely the same. But in real world, I believe that the parameters can be shared. It is just a limitation of the simulation software and can be dealt with. When the start point is changed, the time of the ***passive_wait()*** is also supposed to change.

9.1.2.3 GRASPING AND RELEASING

The main function is the grasping and releasing progress which is conducted by the codes as follows,

```
int a = 0;
while (a<1) {
arm_increase_height();
gripper_release();
passive_wait(3.0);
arm_increase_height();
passive_wait(2.0);
gripper_grip();
passive_wait(2.0);
arm_decrease_height();
passive_wait(3.0);
arm_increase_orientation();
arm_increase_orientation();
arm_increase_orientation();
arm_increase_orientation();
passive_wait(3.0);
arm_increase_height();
passive_wait(3.0);
gripper_release();
passive_wait(2.0);
arm_reset();
gripper_grip();
a=2;
}
```

Figure57 – UR3e, 5e and 10e

The ***passive_wait()*** also plays an important role in main function. It is used to make the arm wait for the previous order to complete then conduct the next one and it is not replaceable. When the position of the food, or the distance between the vehicle and pond change, the whole code needs to be adjusted. Extra orders are needed or the existing code are supposed to delete, depending on the situations. Another solution is to control the robot using keyboard. This requires extra header which is ***#include <webots/keyboard.h>***, and some extra codes are needed:

```
int c = wb_keyboard_get_key();
if ((c >= 0) && c != pc) {
  switch (c) {
    case ' ':
      printf("Reset\n");
      arm_reset();
      break;
    case '+':
    case 388:
    case 65585:
      printf("Grip\n");
      gripper_grip();
      break;
    case '-':
    case 390:
      printf("Ungrip\n");
      gripper_release();
      break;
    case 332:
    case WB_KEYBOARD_UP | WB_KEYBOARD_SHIFT:
      printf("Increase arm height\n");
      arm_increase_height();
      break;
    case 326:
    case WB_KEYBOARD_DOWN | WB_KEYBOARD_SHIFT:
      printf("Decrease arm height\n");
      arm_decrease_height();
      break;
    case 330:
    case WB_KEYBOARD_RIGHT | WB_KEYBOARD_SHIFT:
      printf("Increase arm orientation\n");
      arm_increase_orientation();
      break;
    case 328:
    case WB_KEYBOARD_LEFT | WB_KEYBOARD_SHIFT:
      printf("Decrease arm orientation\n");
      arm_decrease_orientation();
      break;
    default:
      fprintf(stderr, "Wrong keyboard input\n");
      break;
  }
}
```

Figure58- Keyboard-Control solution

The result is tested and proved to be feasible. But in our project, as the whole simulation needs to be finished automatically, this solution is not adopted.

Figure59 - Collision of the Food Container

Another problem is the food container. The 4 baffles are built and they should block the food in task-1 which includes times of turning, and do not affect the turning of the arm. The height and the width of the baffles were modulated time after time and finally the task could be done perfectly.

9.1.3 RESULTS AND DISCUSSION BY YIJUN ZHAO[14]

The details are showed in individual part.

9.2.1 SYSTEM INTEGRATION IN TASK-3[15]

Crossing the bridge and the avoidance of trees contain the building of car model, codes in controller and simulation in ***WEBOT*** platform. Task 3 car model contains the ***hingejoint*** to connect two distance sensors (one in front and one on the left side). The main codes in controller are shown in individual introduction and to get the return value for distance sensor.

9.2.2 RESULT AND DISCUSSION OF TASK-3[16]

In this task, we found that the distance of the sensor detection range is limited, the specific length as shown in figure in red line, only when the red line touches the object distance sensor will be activated, the role of ***wb_distance_sensor_enable*** function is activated, distance sensor in the code TIME_STEP value represents the distance sensor refresh rate, the greater the TIME_STEP, sensors are more sensitive.

```
if (avoid_obstacle_counter > 0) {
  avoid_obstacle_counter--;
  left_speed = -2.1;
  right_speed = 2.1;
  if(avoid_obstacle_counter ==0)
    break;

} else { // read sensors
  double ds_values[1];
  for (i = 0; i < 1; i++)
    ds_values[i] = wb_distance_sensor_get_value(ds[i]);
  if (ds_values[0] <999.0 )
    avoid_obstacle_counter = 57;
}
```

Figure60 – Main code of the Task

[14] Yijun Zhao2357836Z 2017200604030

[15] Jingyuan Feng 2357510F 2017200501036 & Zhuozhao Liu 2357580L 2017200503026

[16] Jingyuan Feng 2357510F 2017200501036 & Zhuozhao Liu 2357580L 2017200503026

The turning of the car is jointly controlled by the speed difference between the wheels on both sides and ***avoid_obstacle_counter***. In other words, the faster the wheel is, the smaller the ***avoid_obstacle_counter*** value is required.

9.3.1 STSTEM INTEGRATION OF TASK-4[17]

HARDWARE: DISTANCE SENSOR

The distance sensor is used in task 4 to identify the arch bridge. When the trolley approaches an obstacle (arch), the distance sensor rays can identify the obstacle (arch).

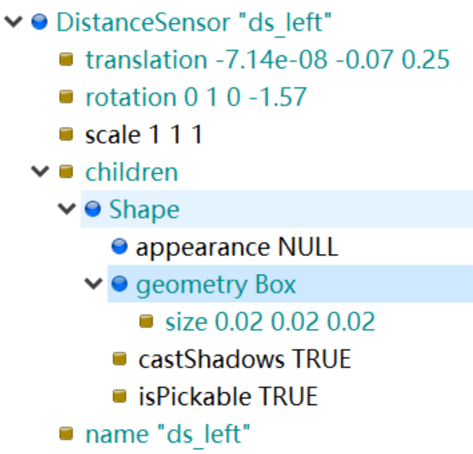


Figure61 – Nodes Integration

9.3.2 DISCUSSION AND RESULTS OF TASK-4[18]

USE DISTANCE SENSORS OR THE CAMERA

Our plan is to recognize the arch bridge through the distance sensor and turn left through the arch bridge. It took us a while to optimize the accuracy of the recognition. At first, our sensor was located at the rear of the car at a distance away from the body. Although this can achieve the recognition of the arch bridge and turn smoothly, the position of distance sensors may affect subsequent experiments. Therefore, we thought about change the distance sensor into the camera, but due to the difficulty of controlling the camera and the advantages of the distance sensor,(such as identifying the color and shape of the arch bridge), we eventually chose to continue to use the sensor to identify the arch bridge. Then, we optimized the code, and finally, the

[17] Yuxiao Luo 2357499l 2017200501025 & Xingyue Wu 2357561W 2017200503007

[18] Yuxiao Luo 2357499l 2017200501025 & Xingyue Wu 2357561W 2017200503007

distance sensor is located between the two tires on the left. The new design helps us successfully complete task-4.

AVOID EXTRA TURNS

Once the left ray starts to detect, it will continually detect the arch. In other word, the robot will turn left constantly and get stuck on the arch.

Demand 1 cycle 4 sum Counters Record position Parallel control slope.

This is related to task3, there is the first and second turn in task3, and the third turn is involved in task four

Then there are four counters

```
bool avoid_obstacle_counter1 = 0;
bool avoid_obstacle_counter2 = 0;
bool avoid_obstacle_counter3 = 0;
bool avoid_obstacle_counter4 = 0;
```

Figure62 – Four Counters

The first turn condition is that the ***counter 1>0 3=0 4=0,***

Then after turning once, the ***counter 4*** becomes 1,

Then the second turn is ***counter 2>0,***

The third time is the ***counter 1>0 4=0,***

Then ‘***break***’.

TURN 90 DEGREES

Discuss the method to turn 90 degrees and go cross the arch.

```
if (ds_values[0] <999.0 )
  avoid_obstacle_counter = 57;
```

```
bool avoid_obstacle_counter1 = 0;
bool avoid_obstacle_counter2 = 0;
bool avoid_obstacle_counter3 = 0;
bool avoid_obstacle_counter4 = 0;
```

Figure63 – Code for Turning

When the value of ***avoid_obstacke_counter*** decreases from 57 to 0, the car will turn 90 degrees.

9.4.1 DISCUSSION AND RESULTS OF TASK-4[19]

HARDWARE: CAMERA IN TASK 5

Camera in task 5 is used to recognize color of colored box on the black line to choose one of three colored lines according the recognized color. It basically plays the same work as it in task 2 but with the forward captured direction.

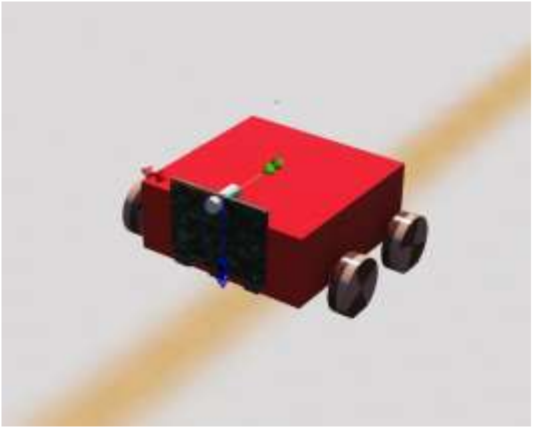


Figure64 – Hardware Overview

CODE PART

```
Camera {
   translation 0 0.3 0
   rotation 0 -1 0 -1.5708153071795863
   children [
     Transform {
       translation 0 -0.18 -0.23
       rotation 1 0 0 1.5708
       children [
         Shape {
           appearance PBRAppearance {
             metalness 0
           }
```

[19] Yimin Tian 2357806T 2017200603010

```
        geometry Cylinder {
          height 0.1
          radius 0.02
        }
      }
    ]
  }
]
recognition Recognition {
}
}
```

9.4.2 DISCUSSION: HOW TO RECOGNIZE RED, YELLOW, PURPLE BOX?[20]

One part of Task 5 is to choose one of three colored line to do line following according to the colored box.

When car, in the line following status, reaches colored box, central process unit is requested to analyze frame captured by camera at this moment and to recognize color for next step.

But how to determine present frame do contain a colored box?

One way is calculating RGB pixels and find RGB relation for each colored boxes.

The use of HSV as a color space for the visual system mitigates the effects of site lighting changes on the visual system.

For strategy one, we use ***wb_camera_get_width wb_camera_get_height*** to get present frame size. With the help of ***wb_camera_image_get_red wb_camera_image_get_green wb_camera_image_get_blue***, pixels of red, green and blue can be directly count out.

[20] Yimin Tian 2357806T 2017200603010

1. For frame 1

Red	Green	Blue
64321	58271	64321

Table7 – RGB for Frame 1

2. For frame 2

Red	Green	Blue
65443	48576	65702

Table8 – RGB for Frame 2

3. For frame 3

Red	Green	Blue
69145	21183	70329

Table9 – RGB for Frame 3

Consider the multiplicative relationship of these three colors, it is easy to say when both red and blue pixels is three times than green one, camera find a purple box. Which write in C language is:

Purple

if ((red > 3 * green) && (blue > 3 * green))

With same procedures, other two-colored boxes can be recognized by:

Red

if ((red > 3 * green) && (red > 3 * blue))

Yellow

if ((red > 3 * blue) && (green > 3 * blue))

As original RGB model can handle these three colored boxes very well, more complex HSV color space will not be applied in Task-5. Task-5 will use ***wb_camera_get_width*** ***wb_camera_get_height*** to get present frame size and use ***wb_camera_image_get_red*** ***wb_camera_image_get_green*** ***wb_camera_image_get_blue***, pixels of red, green and blue to count each color pixels out. With help of the recognition conditions, camera module has ability to distinguish whether present frame contains a colored box or not.

10. Conclusions

Task 1 is a line following task. I focused on how to choose the best approach to make the process efficient, and I think it is a significant ability for engineer is to find problem and solve it. That is what I did in task 1 to determine the best parameters and structure of robot.[21]

For the mechanical arm part, as the controller is apart from the main controller, the sensor parameters cannot be shared by two main functions either. So that an extra camera is supposed to add in real situation. In our project, as the simulation is ideal and more than little modification between different start point in task-1, the trigger can be replaced by a ***wait()*** function. Though I still write the control command into a function to be called by color detection mode.[22] For Task-2, the difficulty of this part is color detecting, so I spend most of time to study the function of camera. After that the movement of the car is just the basic use of program.[23]

Building the car and devices model is the first step. The more crucial part is the codes modification and run. The value for ***avoid_counter*** we find that 57 is a suitable case for turning 90 degrees and while function to connect different distance sensor to work by defining the different counter to initial value=0.Only by trying again and again that makes a success.[24]

Through the final construction of the robot car, our robot car can complete task 4. Besides, it can be well connected with task 3 and task 5. In the final assessment, it successfully completed crossing the bridge and avoided the trees.[25]

Through whole task 5, we have given simple solution to solve colored line following and compared two color space to choose RGB as color recognition plan. For completion, camera with a white cylinder shape is placed in the front of the car and,

[21] Qingyun Wang 2357811W 2017200603015

[22] Weixi Xiong 2357797X 2017200603001

[23] Yijun Zhao2357836Z 2017200604030

[24] Jingyuan Feng 2357510F 2017200501036 & Zhuozhao Liu 2357580L 2017200503026

[25] Yuxiao Luo 2357499l 2017200501025 & Xingyue Wu 2357561W 2017200503007

in code part, two loops, where one is for black line following and another is for recognition color as well as colored line following, are connected with “if(((red > 3 * green) && (red > 3 * blue)) || ((red > 3 * green) && (blue > 3 * green)) || ((red > 3 * blue) && (green > 3 * blue)))”.[26]

[26] Wentao Du 2357813D 2017200603017 & Yimin Tian 2357806T 2017200603010 & Mianzhe Wu 2289329W 2016200304028

11. References

[1] “KUKA’s youBot MOBILE ROBOTIC “[Online]. Available: http://www.cyberbotics.com/doc/guide/youbot [Accessed 20 6 2020].

12. Appendices

TOTAL FINISHED PROGRAM

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <webots/camera.h>
#include <webots/motor.h>
#include <webots/robot.h>
#include <webots/utils/system.h>
#include <math.h>




#include <webots/distance_sensor.h>
#include <webots/led.h>
#include <webots/light_sensor.h>

// Global defines
#define TRUE 1
#define FALSE 0
#define NO_SIDE -1
#define LEFT 0
#define RIGHT 1
#define WHITE 0
#define BLACK 1
#define SIMULATION 0  // for wb_robot_get_mode() function
```

```
#define REALITY 2     // for wb_robot_get_mode() function
#define TIME_STEP 32  // [ms]

// Turing point
#define GEN_SPEED 500
#define NO_SPEED 0

// 3 IR ground color sensors
#define NB_GROUND_SENS 3
#define GS_WHITE 900
#define GS_LEFT 0
#define GS_CENTER 1
#define GS_RIGHT 2
WbDeviceTag gs[NB_GROUND_SENS]; /* ground sensors */
unsigned short gs_value[NB_GROUND_SENS] = {0, 0, 0};


WbDeviceTag wheels[4];
//------------------------------------------------------------------------------
//
//    BEHAVIORAL MODULES
//
//------------------------------------------------------------------------------

////////////////////////////////////////////
// LFM - Line Following Module
//
// This module implements a very simple, Braitenberg-like behavior in order
```

```
// to follow a black line on the ground. Output speeds are stored in
// lfm_speed[LEFT] and lfm_speed[RIGHT].

int lfm_speed[2];

#define LFM_FORWARD_SPEED 3000
#define LFM_K_GS_SPEED 3.5

void LineFollowingModule(void) {
  int DeltaS = gs_value[GS_RIGHT] - gs_value[GS_LEFT];

  lfm_speed[LEFT] = LFM_FORWARD_SPEED - LFM_K_GS_SPEED *DeltaS;
  lfm_speed[RIGHT] = LFM_FORWARD_SPEED + LFM_K_GS_SPEED *DeltaS;
}
void TurningRightModule(void){
  lfm_speed[LEFT] = 0.97*LFM_FORWARD_SPEED;
lfm_speed[RIGHT] = -1*LFM_FORWARD_SPEED;
}

//TASK 5 LINEFOLLOWING
#define LFM_FORWARD_SPEED1 2000
#define LFM_K_GS_SPEED1 13.5

void LineFollowingModule1(void) {
  int DeltaS = gs_value[GS_RIGHT] - gs_value[GS_LEFT];

  lfm_speed[LEFT] = LFM_FORWARD_SPEED1 - LFM_K_GS_SPEED1 *DeltaS;
  lfm_speed[RIGHT] = LFM_FORWARD_SPEED1 + LFM_K_GS_SPEED1 *DeltaS;
```

```
}


/*void step(double seconds) {
  const double ms = seconds * 1000.0;
  int elapsed_time = 0;
  while (elapsed_time < ms) {
        wb_robot_step(time_step);
        elapsed_time += time_step;
  }
}*/
static void step() { //define function step
  if (wb_robot_step(TIME_STEP) == -1) {
        wb_robot_cleanup();
        exit(EXIT_SUCCESS);
  }
}
static void passive_wait(double sec) {
  double start_time = wb_robot_get_time();
  do {
        step();
  } while (start_time + sec > wb_robot_get_time());
}
//----------------------------------------------------------------------------
//
//    CONTROLLER
//
//----------------------------------------------------------------------------
```

```
//////////////////////////////////////////
// Main
int main() {
  int width, height;
  int i, j;
  int speed[2];
  int red,green,blue;
  WbDeviceTag camera;
  /* intialize Webots */
  wb_robot_init();

  /* initialization */
        char name[20];
  for (i = 0; i < NB_GROUND_SENS; i++) {
        sprintf(name, "gs%d", i);
        gs[i] = wb_robot_get_device(name); /* ground sensors */
        wb_distance_sensor_enable(gs[i], TIME_STEP);
  }
  // motors

  WbDeviceTag wheels[4];
  char wheels_names[4][20] = {"Left_Front_wheel", "Right_Front_wheel",
"Left_Behind_wheel", "Right_Behind_wheel"};
  for (i = 0; i < 4; i++) {
        wheels[i] = wb_robot_get_device(wheels_names[i]);
        wb_motor_set_position(wheels[i], INFINITY);
  }
        wb_motor_set_velocity(wheels[0],0);
```

```
        wb_motor_set_velocity(wheels[1],0);
        wb_motor_set_velocity(wheels[2],0);
        wb_motor_set_velocity(wheels[3],0);

        camera = wb_robot_get_device("camera");
        wb_camera_enable(camera, TIME_STEP);
        width = wb_camera_get_width(camera);
        height = wb_camera_get_height(camera);

    /* Main loop */
    while (wb_robot_step(TIME_STEP) != -1) {
        /* Get the new camera values */
        const unsigned char *image = wb_camera_get_image(camera);

        /* Reset the sums */
        red = 0;
        green = 0;
        blue = 0;

        for (i = width / 3; i < 2 * width / 3; i++) {
          for (j = height / 2; j < 3 * height / 4; j++) {
                red += wb_camera_image_get_red(image, width, i, j);
                blue += wb_camera_image_get_blue(image, width, i, j);
                green += wb_camera_image_get_green(image, width, i, j);
          }
        }
        //if (red>200&&green>200&&blue<50){
        if ((red > 4 * blue) && (green > 4 * blue)){
```

```
        i=0;
    wb_robot_step(TIME_STEP);
        speed[LEFT]=0 ;
        speed[RIGHT]=0;
        TurningRightModule();
        speed[LEFT] = lfm_speed[LEFT];
        speed[RIGHT] = lfm_speed[RIGHT];
        wb_motor_set_velocity(wheels[0], 0.00628 *speed[LEFT]);
        wb_motor_set_velocity(wheels[1], 0.00628 *speed[RIGHT]);
        wb_motor_set_velocity(wheels[2], 0.00628 *speed[LEFT]);
        wb_motor_set_velocity(wheels[3], 0.00628 *speed[RIGHT]);
    printf("i find a yellow box\n");
          passive_wait(0.60);

        wb_motor_set_velocity(wheels[0], 10);
        wb_motor_set_velocity(wheels[1], 10);
        wb_motor_set_velocity(wheels[2], 10);
        wb_motor_set_velocity(wheels[3], 10);
        passive_wait(1.8);

    wb_motor_set_velocity(wheels[0], 0);
        wb_motor_set_velocity(wheels[1], 0);
        wb_motor_set_velocity(wheels[2], 0);
        wb_motor_set_velocity(wheels[3], 0);
          passive_wait(22);
          wb_motor_set_velocity(wheels[0], -10);
        wb_motor_set_velocity(wheels[1], -10);
        wb_motor_set_velocity(wheels[2], -10);
```

```
            wb_motor_set_velocity(wheels[3], -10);
            passive_wait(1.5);
            TurningRightModule();
            speed[LEFT] = lfm_speed[RIGHT];
            speed[RIGHT] = lfm_speed[LEFT];
            wb_motor_set_velocity(wheels[0], 0.00628 *speed[LEFT]);
            wb_motor_set_velocity(wheels[1], 0.00628 *speed[RIGHT]);
            wb_motor_set_velocity(wheels[2], 0.00628 *speed[LEFT]);
            wb_motor_set_velocity(wheels[3], 0.00628 *speed[RIGHT]);
            passive_wait(0.4481);
             wb_motor_set_velocity(wheels[0], 15);
            wb_motor_set_velocity(wheels[1], 15);
            wb_motor_set_velocity(wheels[2], 15);
            wb_motor_set_velocity(wheels[3], 15);
              passive_wait(1);



    break;
   }
    for (;;) {  // Main loop
            // Run one simulation step
            wb_robot_step(TIME_STEP);

            for (i = 0; i < NB_GROUND_SENS; i++)
              gs_value[i] = wb_distance_sensor_get_value(gs[i]);

            // Speed initialization
```

```
            speed[LEFT]=0 ;
            speed[RIGHT]=0;

            // *** START OF SUBSUMPTION ARCHITECTURE ***

            // LFM - Line Following Module
            LineFollowingModule();

            speed[LEFT] = lfm_speed[LEFT];
            speed[RIGHT] = lfm_speed[RIGHT];


            // *** END OF SUBSUMPTION ARCHITECTURE ***


            // Set wheel speeds

            wb_motor_set_velocity(wheels[0], 0.00628 *speed[LEFT]);
            wb_motor_set_velocity(wheels[1], 0.00628 *speed[RIGHT]);
            wb_motor_set_velocity(wheels[2], 0.00628 *speed[LEFT]);
            wb_motor_set_velocity(wheels[3], 0.00628 *speed[RIGHT]);
            break;

    }
    }


    bool avoid_obstacle_counter1 = 0;
```

```
    bool avoid_obstacle_counter2 = 0;
    bool avoid_obstacle_counter3 = 0;
    bool avoid_obstacle_counter4 = 0;
    WbDeviceTag ds[1];
    char ds_names[1][10] = {"ds_left"};
    for (i = 0; i < 1; i++) {
        ds[i] = wb_robot_get_device(ds_names[i]);
        wb_distance_sensor_enable(ds[i], TIME_STEP);
    }
    WbDeviceTag ds_r[1];
    char ds_rnames[1][10] = {"ds_right"};
    for (i = 0; i < 1; i++) {
        ds_r[i] = wb_robot_get_device(ds_rnames[i]);
        wb_distance_sensor_enable(ds_r[i], TIME_STEP);
    }

    while (wb_robot_step(TIME_STEP) != -1) {
        double left_speed = 5;
        double right_speed = 5;
        if (avoid_obstacle_counter1 > 0&& avoid_obstacle_counter3 == 0&&avoid_obstacle_counter4==0)
         {
          avoid_obstacle_counter1--;
          left_speed = -2.1;
          right_speed = 2.1;
          if(avoid_obstacle_counter1==0)
          {avoid_obstacle_counter4=1; }}
        else if ((avoid_obstacle_counter1 > 0)&&(avoid_obstacle_counter4 ==1)) {
          avoid_obstacle_counter1--;
```

```
            left_speed = -2.1;
            right_speed = 2.1;
            if(avoid_obstacle_counter1 ==0)
            break;
           }

    else if (avoid_obstacle_counter2 > 0) {
            avoid_obstacle_counter2--;
            left_speed = 2.1;
            right_speed = -2.1;
            avoid_obstacle_counter3=1;
            if(avoid_obstacle_counter2==0)
            {left_speed = 5;
            right_speed = 5;
            wb_motor_set_velocity(wheels[0], left_speed);
            wb_motor_set_velocity(wheels[1], right_speed);
            wb_motor_set_velocity(wheels[2], left_speed);
            wb_motor_set_velocity(wheels[3], right_speed);
            passive_wait(6);
            avoid_obstacle_counter3=0;
            }
           }
           else{ // read sensors
            double ds_values1[1];
            for (i = 0; i < 1; i++)
                    ds_values1[i] = wb_distance_sensor_get_value(ds[i]);
            if (ds_values1[0] <999.0 )
                    {passive_wait(1);
```

```
                    avoid_obstacle_counter1 = 107;}
            double ds_values2[1];
            for (i = 0; i < 1; i++)
                    ds_values2[i] = wb_distance_sensor_get_value(ds_r[i]);
            if (ds_values2[0] <999.0 )
                    avoid_obstacle_counter2 = 117 ;
           }
           wb_motor_set_velocity(wheels[0], left_speed);
           wb_motor_set_velocity(wheels[1], right_speed);
           wb_motor_set_velocity(wheels[2], left_speed);
           wb_motor_set_velocity(wheels[3], right_speed);
  }

double left_speed = 5;//接巡线
double right_speed = 5;
wb_motor_set_velocity(wheels[0], left_speed);
wb_motor_set_velocity(wheels[1], right_speed);
wb_motor_set_velocity(wheels[2], left_speed);
wb_motor_set_velocity(wheels[3], right_speed);
//task5
 WbDeviceTag task5;
 int current_blob;
 /* Get the camera device, enable it, and store its width and height */
  task5 = wb_robot_get_device("task5");
  wb_camera_enable(task5, TIME_STEP);
  width = wb_camera_get_width(task5);
  height = wb_camera_get_height(task5);

```

```
/* Main loop */
while (wb_robot_step(TIME_STEP) != -1) {


        // line follow
        while(1) {

        // continuously dectect color box
        const unsigned char *image = wb_camera_get_image(task5);
        red = 0;
        green = 0;
        blue = 0;
        for (i = width / 3; i < 2 * width / 3; i++) {
          for (j = height / 2; j < 3 * height / 4; j++) {
                red += wb_camera_image_get_red(image, width, i, j);
                blue += wb_camera_image_get_blue(image, width, i, j);
                green += wb_camera_image_get_green(image, width, i, j);
          }
        }
        // print values in console
        printf("red value is %d .\n", red);
        printf("blue value is %d .\n", blue);
        printf("green value is %d .\n", green);


        // begin line follow
        wb_robot_step(TIME_STEP);

```

```
		for (i = 0; i < NB_GROUND_SENS; i++)
			gs_value[i] = wb_distance_sensor_get_value(gs[i]);

		speed[LEFT]=0 ;
		speed[RIGHT]=0;
		LineFollowingModule();
		speed[LEFT] = lfm_speed[LEFT];
		speed[RIGHT] = lfm_speed[RIGHT];
		wb_motor_set_velocity(wheels[0], 0.00628 *speed[LEFT]);
		wb_motor_set_velocity(wheels[1], 0.00628 *speed[RIGHT]);
		wb_motor_set_velocity(wheels[2], 0.00628 *speed[LEFT]);
		wb_motor_set_velocity(wheels[3], 0.00628 *speed[RIGHT]);
		// end line follow

		/* detect color box,
		 * stop,
		 * waiting next step.
		 */
		if(((red > 3 * green) && (red > 3 * blue)) || ((red > 3 * green) && (blue > 3 * green)) || ((red > 3 * blue) && (green > 3 * blue))){
			wb_motor_set_velocity(wheels[0], 0.00628 *speed[LEFT]);
			wb_motor_set_velocity(wheels[1], 0.00628 *speed[RIGHT]);
			wb_motor_set_velocity(wheels[2], 0.00628 *speed[LEFT]);
			wb_motor_set_velocity(wheels[3], 0.00628 *speed[RIGHT]);
			break;
		}
		}

		/*
```

```
         * If a component is much more represented than the other ones, a color box is detected
         * next step is choose a direction,
         * then, line follow
         */
        if ((red > 3 * green) && (red > 3 * blue)){
         current_blob = 1;//red
         printf("Looks like I found a %d blob.\n", current_blob);
                for (;;) {  // Main loop
                // Run one simulation step
                wb_robot_step(TIME_STEP);
                for (i = 0; i < NB_GROUND_SENS; i++)
                  gs_value[i] = wb_distance_sensor_get_value(gs[i]);

                // Speed initialization
                speed[LEFT]=0 ;
                speed[RIGHT]=0;
                LineFollowingModule1();
                speed[LEFT] = lfm_speed[LEFT];
                speed[RIGHT] = lfm_speed[RIGHT];
                wb_motor_set_velocity(wheels[0], 0.00628 *speed[LEFT]);
                wb_motor_set_velocity(wheels[1], 0.00628 *speed[RIGHT]);
                wb_motor_set_velocity(wheels[2], 0.00628 *speed[LEFT]);
                wb_motor_set_velocity(wheels[3], 0.00628 *speed[RIGHT]);
         }

        }

        else if ((red > 3 * green) && (blue > 3 * green)){
```

```
current_blob = 2;//purple
printf("Looks like I found a %d blob.\n", current_blob);

// after detected box, keep moving for a distance
int a=0;
while ((wb_robot_step(TIME_STEP) != -1) & (a<= 145)){
speed[LEFT]= GEN_SPEED;
speed[RIGHT]= GEN_SPEED;
wb_motor_set_velocity(wheels[0], 0.01628 *speed[LEFT]);
wb_motor_set_velocity(wheels[1], 0.01628 *speed[RIGHT]);
wb_motor_set_velocity(wheels[2], 0.01628 *speed[LEFT]);
wb_motor_set_velocity(wheels[3], 0.01628 *speed[RIGHT]);
a = a+1;
}

wb_motor_set_velocity(wheels[0], 0);
wb_motor_set_velocity(wheels[1], 0);
wb_motor_set_velocity(wheels[2], 0);
wb_motor_set_velocity(wheels[3], 0);

// turning point
a=0;
while ((wb_robot_step(TIME_STEP) != -1) & (a<= 80)){
speed[LEFT]= GEN_SPEED;
speed[RIGHT]= NO_SPEED;
wb_motor_set_velocity(wheels[0], 0.00628 *speed[LEFT]);
wb_motor_set_velocity(wheels[1], 0.00628 *speed[RIGHT]);
wb_motor_set_velocity(wheels[2], 0.00628 *speed[LEFT]);
```

```
        wb_motor_set_velocity(wheels[3], 0.00628 *speed[RIGHT]);
        a = a+1;
        }

        // reset velocity to 0
        wb_motor_set_velocity(wheels[0], 0 *speed[LEFT]);
        wb_motor_set_velocity(wheels[1], 0 *speed[RIGHT]);
        wb_motor_set_velocity(wheels[2], 0 *speed[LEFT]);
        wb_motor_set_velocity(wheels[3], 0 *speed[RIGHT]);

        // line following
        for (;;) {  // Main loop
                // Run one simulation step
                wb_robot_step(TIME_STEP);
                for (i = 0; i < NB_GROUND_SENS; i++)
                  gs_value[i] = wb_distance_sensor_get_value(gs[i]);

                // Speed initialization
                speed[LEFT]=0 ;
                speed[RIGHT]=0;
                LineFollowingModule1();
                speed[LEFT] = lfm_speed[LEFT];
                speed[RIGHT] = lfm_speed[RIGHT];
                wb_motor_set_velocity(wheels[0], 0.00628 *speed[LEFT]);
                wb_motor_set_velocity(wheels[1], 0.00628 *speed[RIGHT]);
                wb_motor_set_velocity(wheels[2], 0.00628 *speed[LEFT]);
                wb_motor_set_velocity(wheels[3], 0.00628 *speed[RIGHT]);
        }
```

```
        }

        else if ((red > 3 * blue) && (green > 3 * blue)){
          current_blob = 3;//yellow
          printf("Looks like I found a %d blob.\n", current_blob);

           // after detected box, keep moving for a distance
          int a=0;
          while ((wb_robot_step(TIME_STEP) != -1) & (a<= 145)){
          speed[LEFT]= GEN_SPEED;
          speed[RIGHT]= GEN_SPEED;
          wb_motor_set_velocity(wheels[0], 0.01628 *speed[LEFT]);
          wb_motor_set_velocity(wheels[1], 0.01628 *speed[RIGHT]);
          wb_motor_set_velocity(wheels[2], 0.01628 *speed[LEFT]);
          wb_motor_set_velocity(wheels[3], 0.01628 *speed[RIGHT]);
          a = a+1;
          }

          wb_motor_set_velocity(wheels[0], 0);
          wb_motor_set_velocity(wheels[1], 0);
          wb_motor_set_velocity(wheels[2], 0);
          wb_motor_set_velocity(wheels[3], 0);

          // turning point
          a=0;
          while ((wb_robot_step(TIME_STEP) != -1) & (a<= 80)){
          speed[LEFT]= NO_SPEED;
          speed[RIGHT]= GEN_SPEED;
```

```
        wb_motor_set_velocity(wheels[0], 0.00628 *speed[LEFT]);
        wb_motor_set_velocity(wheels[1], 0.00628 *speed[RIGHT]);
        wb_motor_set_velocity(wheels[2], 0.00628 *speed[LEFT]);
        wb_motor_set_velocity(wheels[3], 0.00628 *speed[RIGHT]);
        a = a+1;
        }

        // reset velocity to 0
        wb_motor_set_velocity(wheels[0], 0 *speed[LEFT]);
        wb_motor_set_velocity(wheels[1], 0 *speed[RIGHT]);
        wb_motor_set_velocity(wheels[2], 0 *speed[LEFT]);
        wb_motor_set_velocity(wheels[3], 0 *speed[RIGHT]);

        // line following
        for (;;) {  // Main loop
                // Run one simulation step
                wb_robot_step(TIME_STEP);
                for (i = 0; i < NB_GROUND_SENS; i++)
                  gs_value[i] = wb_distance_sensor_get_value(gs[i]);

                // Speed initialization
                speed[LEFT]=0 ;
                speed[RIGHT]=0;
                LineFollowingModule1();
                speed[LEFT] = lfm_speed[LEFT];
                speed[RIGHT] = lfm_speed[RIGHT];
                wb_motor_set_velocity(wheels[0], 0.00628 *speed[LEFT]);
                wb_motor_set_velocity(wheels[1], 0.00628 *speed[RIGHT]);
```

```
                wb_motor_set_velocity(wheels[2], 0.00628 *speed[LEFT]);
                wb_motor_set_velocity(wheels[3], 0.00628 *speed[RIGHT]);
          }
         }

         else
          current_blob = 0;//none

  }
wb_robot_cleanup();

  return 0;
}
```

MECHANICAL ARM CODE

```
#include <webots/robot.h>

#include <arm.h>
#include <gripper.h>

#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define TIME_STEP 32

static void step() { //define function step
  if (wb_robot_step(TIME_STEP) == -1) {
```

```
        wb_robot_cleanup();
        exit(EXIT_SUCCESS);
  }
}

static void passive_wait(double sec) {
  double start_time = wb_robot_get_time();
  do {
        step();
  } while (start_time + sec > wb_robot_get_time());
}
static void automatic_behavior() {
  passive_wait(2.0);
  gripper_release();
  arm_set_height(ARM_FRONT_CARDBOARD_BOX);
  passive_wait(4.0);
  gripper_grip();
  passive_wait(1.0);
  arm_set_height(ARM_BACK_PLATE_LOW);
  passive_wait(3.0);
  gripper_release();
  passive_wait(1.0);
  arm_reset();
  passive_wait(5.0);
  gripper_grip();
  passive_wait(1.0);
  passive_wait(1.0);
  gripper_release();
```

```
  arm_set_height(ARM_BACK_PLATE_LOW);
  passive_wait(3.0);
  gripper_grip();
  passive_wait(1.0);
  arm_set_height(ARM_RESET);
  passive_wait(2.0);
  arm_set_height(ARM_FRONT_PLATE);
  arm_set_orientation(ARM_RIGHT);
  passive_wait(4.0);
  arm_set_height(ARM_FRONT_FLOOR);
  passive_wait(2.0);
  gripper_release();
  passive_wait(1.0);
  arm_set_height(ARM_FRONT_PLATE);
  passive_wait(2.0);
  arm_set_height(ARM_RESET);
  passive_wait(2.0);
  arm_reset();
  gripper_grip();
  passive_wait(2.0);
}

int main(int argc, char **argv) {
  wb_robot_init();
  passive_wait(120.0);
  arm_init();
  gripper_init();
  passive_wait(2.0);
```

```
if (argc > 1 && strcmp(argv[1], "de mo") == 0)
 automatic_behavior();
passive_wait(1.0);//改成从起点到转好弯的
int a = 0;
while (a<1) {
arm_increase_height();
gripper_release();
passive_wait(3.0);
arm_increase_height();
passive_wait(2.0);
gripper_grip();
passive_wait(2.0);
arm_decrease_height();
passive_wait(3.0);
arm_increase_orientation();
arm_increase_orientation();
arm_increase_orientation();
arm_increase_orientation();
passive_wait(3.0);
arm_increase_height();
passive_wait(3.0);
gripper_release();
passive_wait(2.0);
arm_reset();
gripper_grip();
a=2;
```

```
    }
    wb_robot_cleanup();

    return 0;
}
```