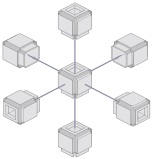
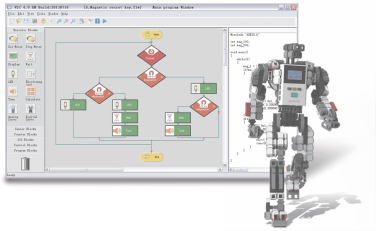
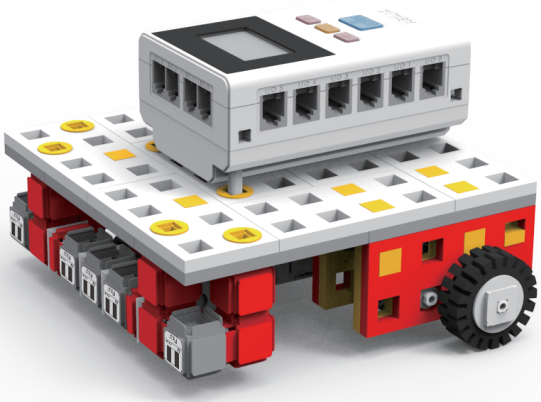
Abilix® Creative Brick Abilix

C203 WER Ability Kit II



Design Guide v1.0





Preface

(C203)WER Ability Kit (Basic) is specially designed for “WER Ability Contest”, namely for building robots to finish stipulated tasks. Although there are various tasks in the contest, the chassis of the robots are mostly built with two-wheeled differential drive structure and 5 grayscale sensors (hereafter referred to as GSS). In light of that, this manual will mainly introduce the building and debugging of the robot.

Before using this book, please read “Ccon102 Manual” and master the basic usage of Ccon102 controller and VJC4.2, among which the “Line Following Blocks” should be paid more attention to.

This book covers six aspects:

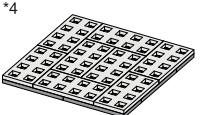
1. The Building Samples of the Line Tracer
2. The Principle of Line Following
3. Program Design of Some Common Routes
4. Suggestions for Debugging
5. FAQ & Solution
6. Bill of Materials

1.The Building Samples of the Line Tracer


Cube
*11

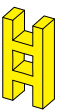

Half cube
*2


Slab(1#)
*4


Slab(8#)
*1


Cube connector
*4


L connector
*10


H connector
*2


Short bolt
*6


Axle(40)
*4


Axle(60)
*2


Gear(12)
*4


Gear(28)
*4

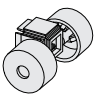

Gear(12/28)
*4


Shaft sleeve
*4

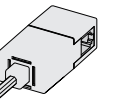

Bearing
*8


Hub
*2


Tire
*2

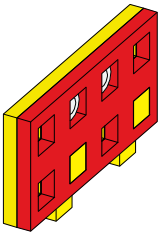

Guide Wheel
*1


Grayscale Sensor
*5


Motor
*2


Controller
*1

1

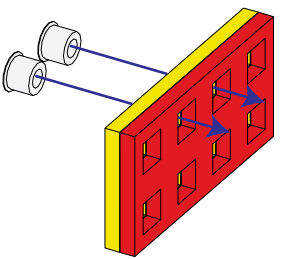


L connector
*2

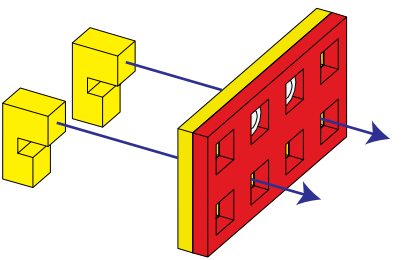
Slab(1#)
*1

Bearing
*2

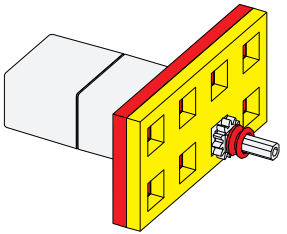
1-1



1-2



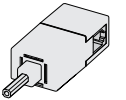
2



Slab(1#)
*1

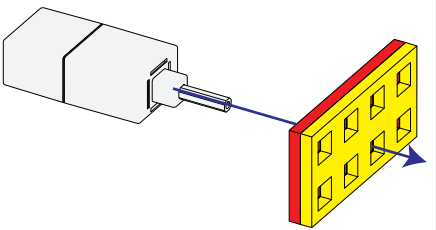
Gear(12)
*1

Shaft sleeve
*1

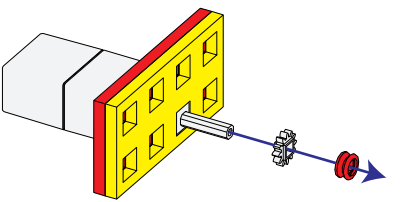


Motor
*1

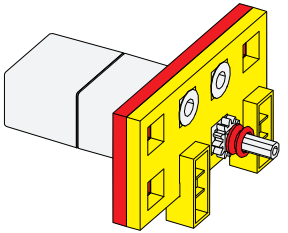
2-1



2-2

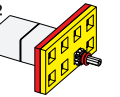


3

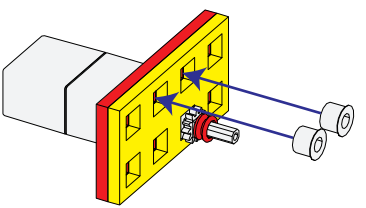


L connector
*2

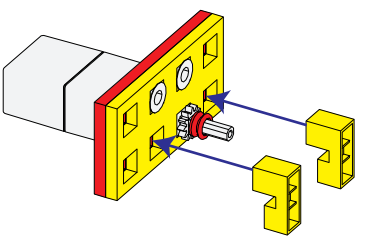
Bearing
*2



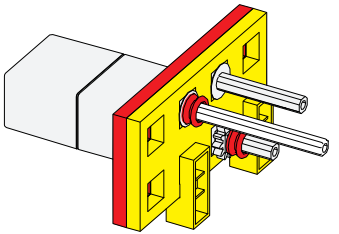
3-1



3-2



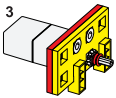
4

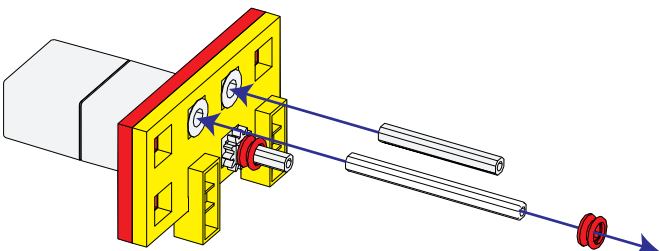


Axle(40)
*1

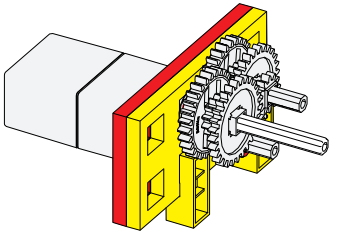
Axle(60)
*1

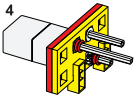
Shaft sleeve
*1

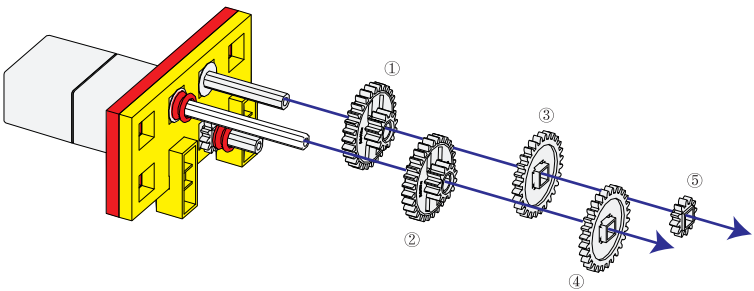




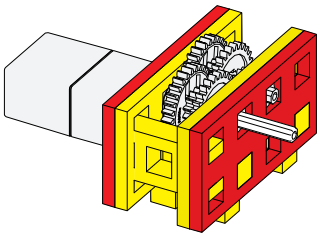
5

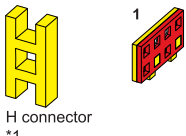


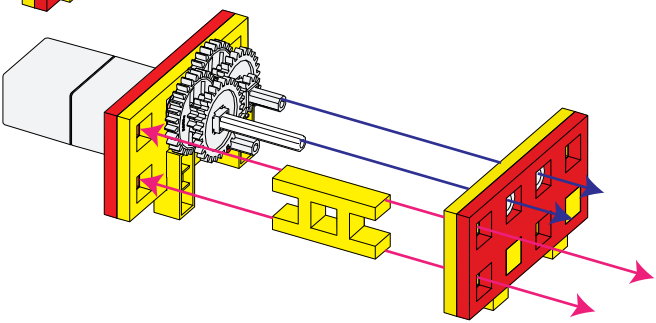




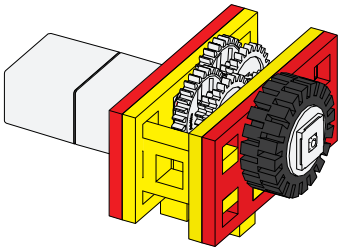
6







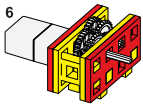
7

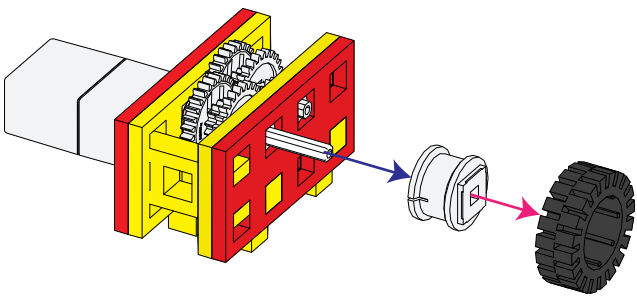


Hub
*1

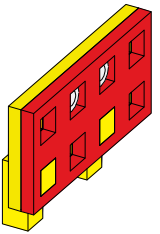
Tire
*1

6





8

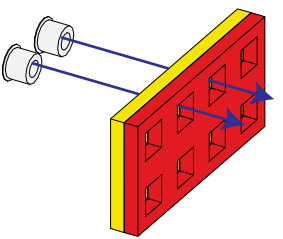


L connector
*2

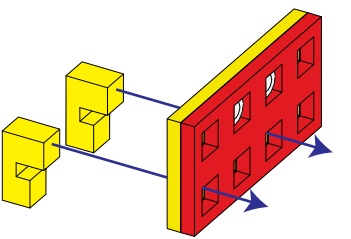
Slab(1#)
*1

Bearing
*2

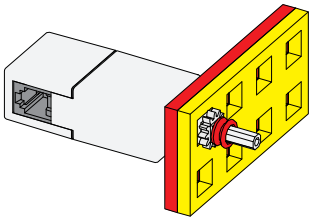
8-1



8-2



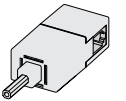
9



Slab(1#)
*1

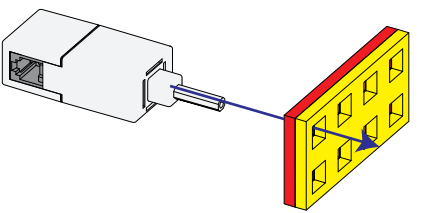
Gear(12)
*1

Shaft sleeve
*1

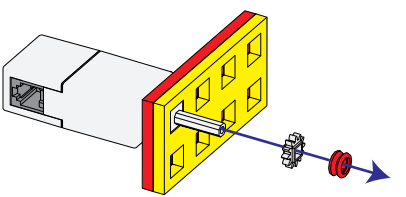


Motor
*1

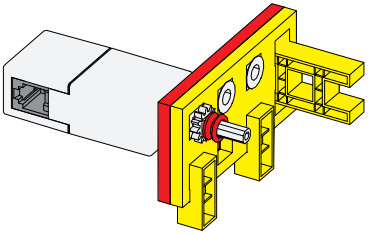
9-1

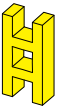


9-2



10

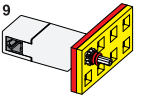




H connector
*1

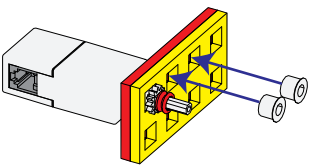
L connector
*2

Bearing
*2

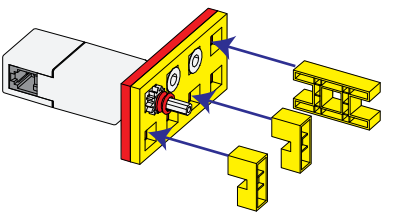


9

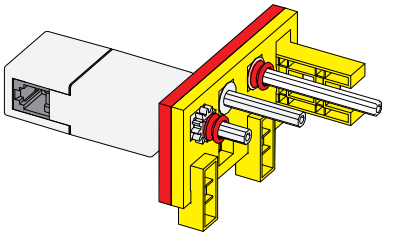
10-1



10-2



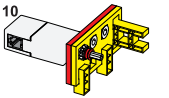
11

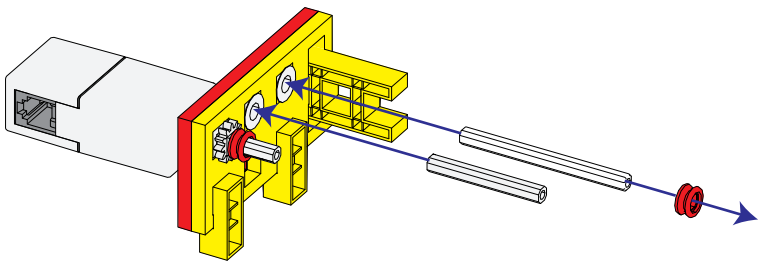


Axle(40)
*1

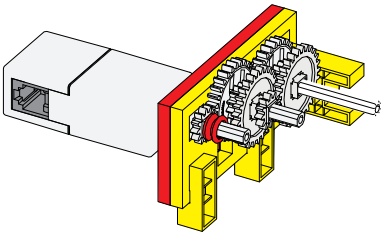
Axle(60)
*1

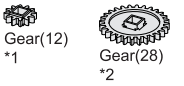
Shaft sleeve
*1

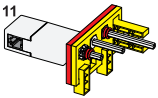


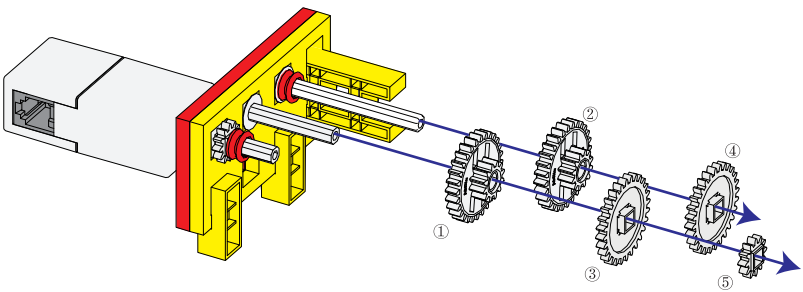


12

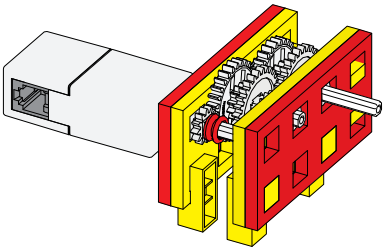


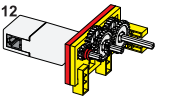


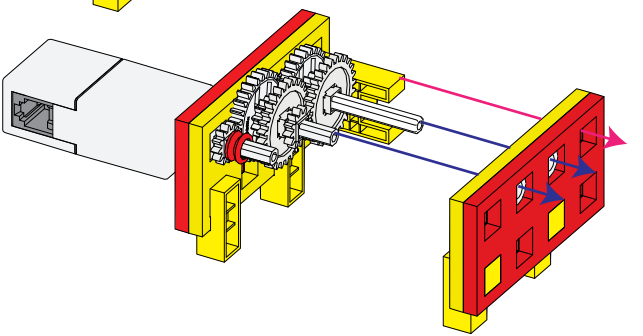




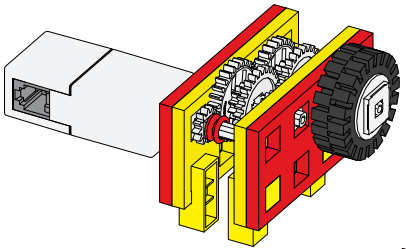
13





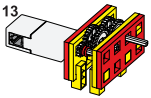


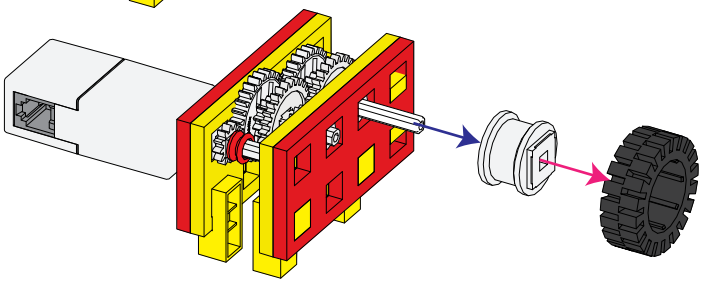
14



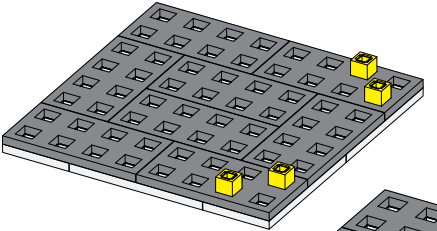
Hub
*1

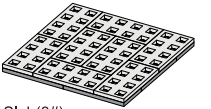
Tire
*1





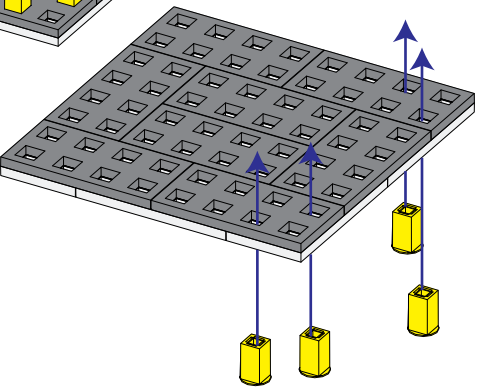
15



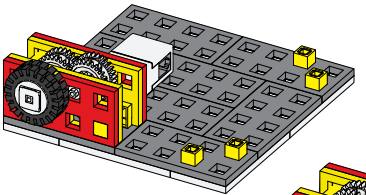


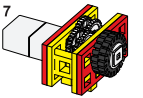
Slab(8#)
*1

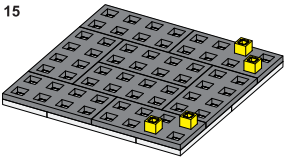
Short bolt
*4

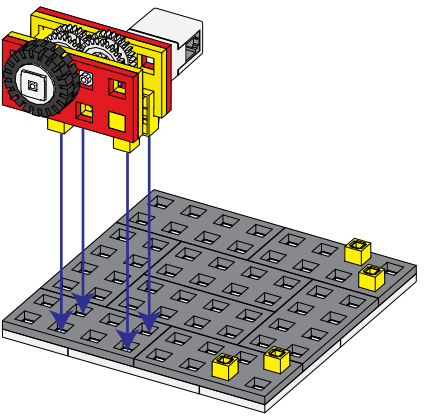


16

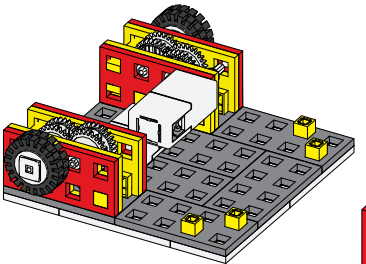




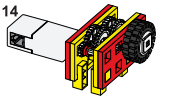




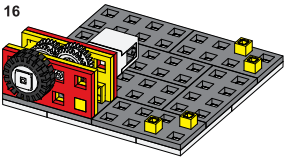
17

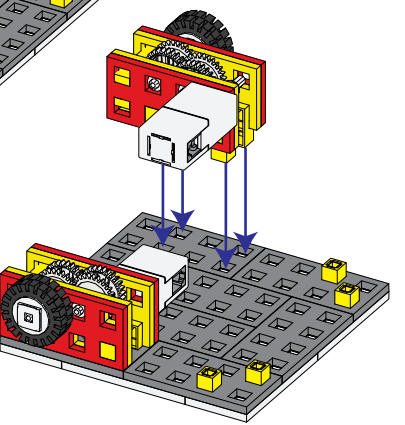


14

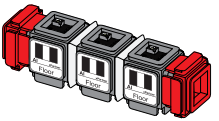


16





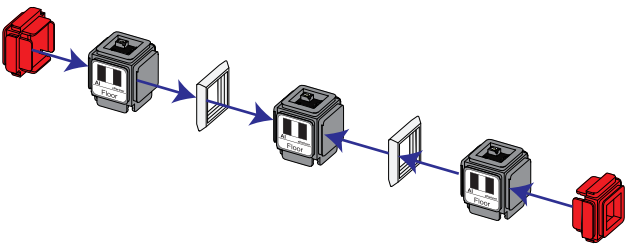
18



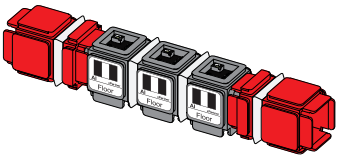
Half cube
*2

Cube connector
*2

Grayscale Sensor
*3



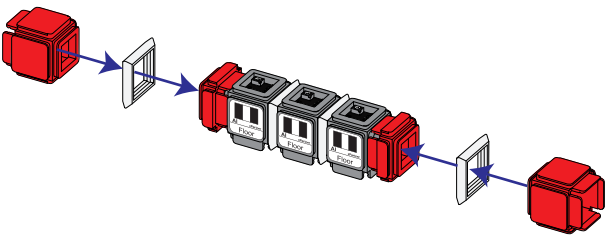
19



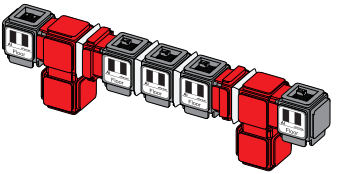
Cube
*2

Cube connector
*2

18



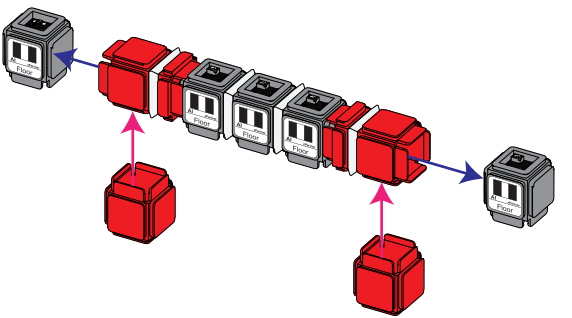
20



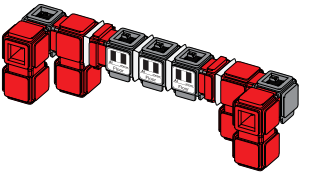
Cube
*2

Grayscale Sensor
*2

19

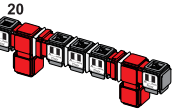


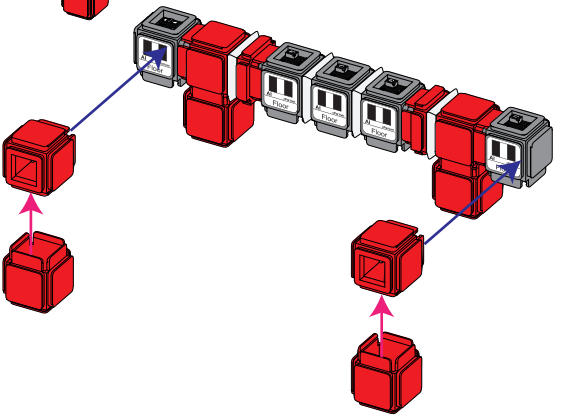
21



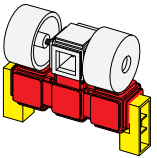
Cube
*4

20





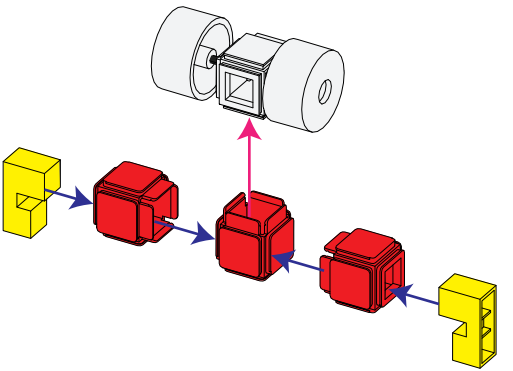
22



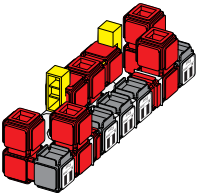
Cube
*3

L connector
*2

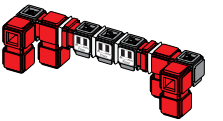
Guide Wheel
*1



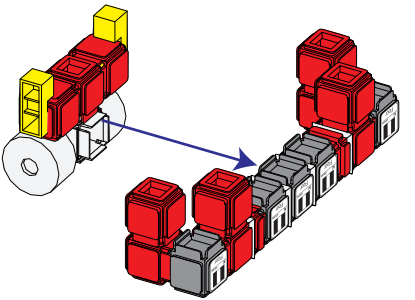
23



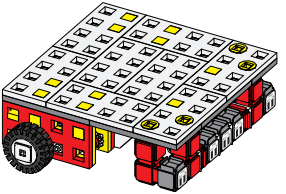
21



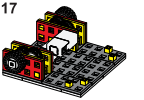
22



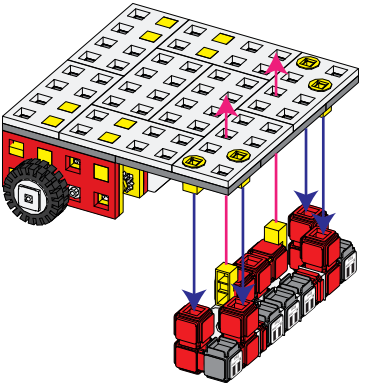
24



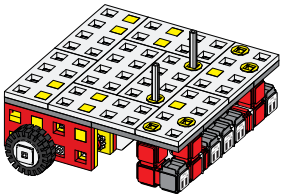
17

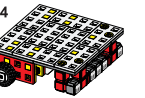


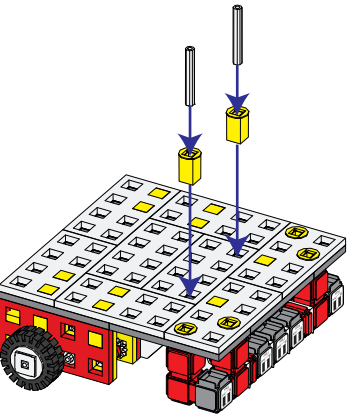
23



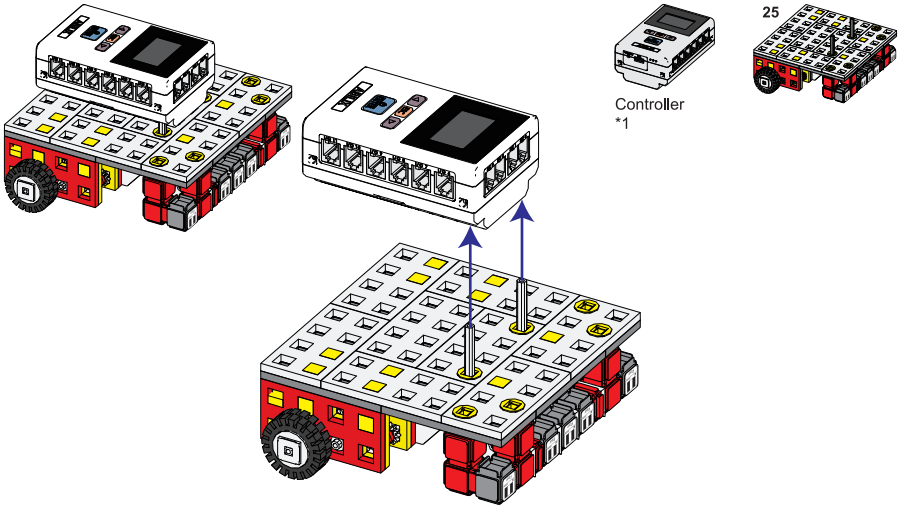
25







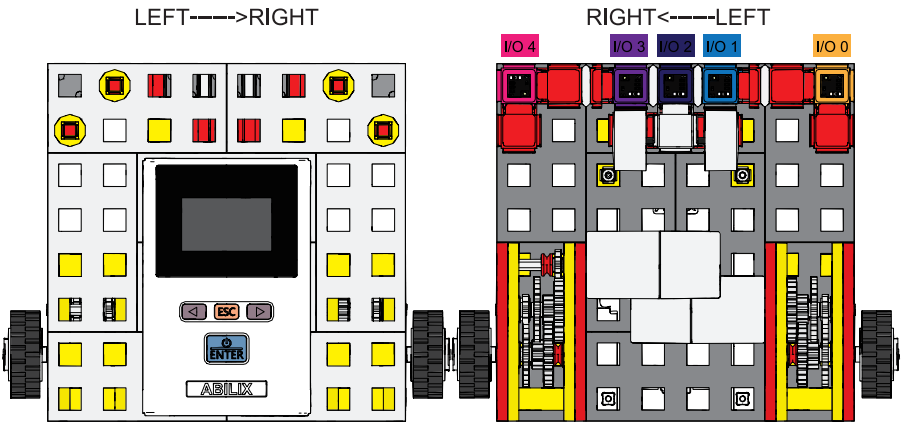
26



27

Wiring

As shown in the picture below left, the GSS are linked to the I/O 0~I/O 4 from left to right, with the left DC motor (hereafter referred to as motor) linked to the DC 0 port, and right one linked to the DC 1 port. The cables are suggested to be bundled with two L connectors.



28

Performance Test of Grayscale Sensor:

1. The arena should be with white background and black line, which can be built with black insulating tape on a white desk.

2. Turn on the controller and enter into the “AI” interface.

3. Put robot’s GSS above the white area, and observe the “AI” interface. The value of I/O 0~I/O 4 should be smaller than 1000 (not 0). If it’s bigger than 1000, the height of the sensors needs to be lowered by attaching cube connector on the 4 cubes, which is used for fixing GSS and connected with the slab. If the reading is 0, then there is a wiring error or something wrong with the GSS.

4. Put robot’s GSS above the black area; observe the “AI” interface, and the value of the I/O 0~I/O 4 should be greater than 3900.

5. Move the robot to the left with your hand so that its GSS can scan from the white ground to the black line and get off from the black line. Observe the screen, you will find the value of 0~4 changing in order, which means the cable is correctly connected. If not, the GSS are perhaps wrongly connected.

29

Performance Test of Motor and Gear

1. Hold the robot; enter into “Motor” interface. With all motors selected, observe the motion state of two wheels after increasing the speed value gradually from 10 to 100. When the speed is 20, both motors run; if not, please check whether the slabs connecting gears are paralleled or the L connectors are correctly attached.

2. When the motor is rotating, wiring status can be checked by observing whether the motor will stop running after unplugging the motor wire of the DC port one by one.

3. Adjust all the motor speed to 100 on the “Motor” interface; put down the robot on the ground; the robot will rotate clockwise.

2.The Principle of Line Following:

This part gives a brief introduction of the principle of line following based on “Line Tracer with 5 GSS”, which will help you to use the “Line following blocks” in VJC4.2.

During line tracing, the two outmost GSS are for recognizing the intersection, with the middle GSS to recognize whether the robot is in the best position (right on the line as the figure A, then the robot can move at full speed.), while the other GSS are to adjust the robot (as shown in the Figure B, C, the robot needs to be adjusted to have its GSS on the line.).

The robot’s motion is achieved by two wheels’ different speed, which refers to wheel’s actual speed instead of setting speed in the program (the control method is called two-wheeled differential drive.). If the left speed is represented with “L”, and the right speed is represented with “R”; then the relation between speed and moving posture can be listed as below:

L＞R: Turn right　L=R: Go straight　　L＜R: Turn left

In conclusion, the robot will turn to the side with lower speed. Therefore, in programming, the side with GSS on the line should reduce its speed, namely the side with GSS on the line should move at a lower speed in line tracing. As for turning, the larger speed difference between two wheels is; the smaller the turning radius is; the bigger the adjusting angle will be. Against such background, the “R-L” value of the line tracer with 5 GSS should suitably set so that the robot will not lose the line.

Figure A

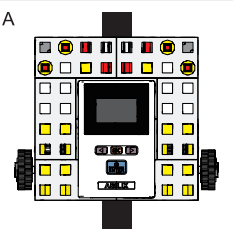


Figure B

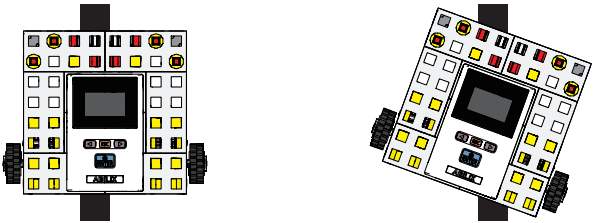
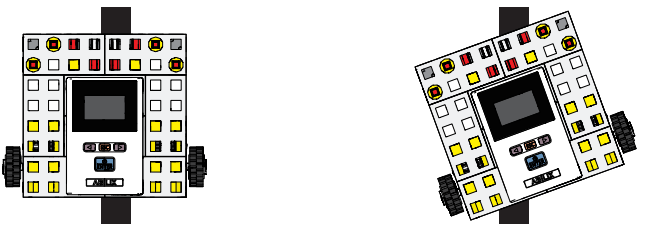


Figure C



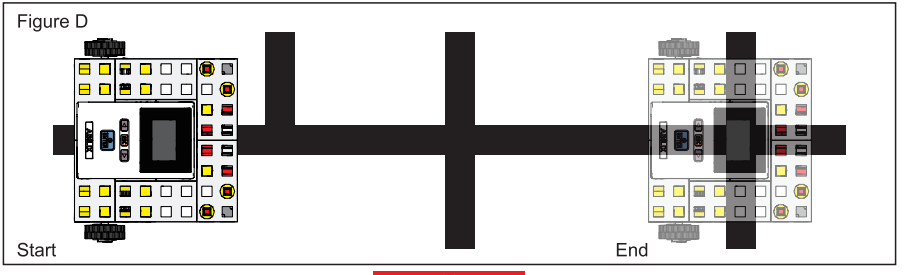
Thus we can write down the code for simple line following principle.

Instruction: The GSS and motor are named after their connected port in step 27 in “Building samples”, e.g. GSS1 stands for the GSS linked to I/O1; motor 1 means the motor linked to the DC1.

IF	GSS1 on the line	DO turn left
ELSE IF	GSS3 on the line	DO turn right
ELSE IF	GSS2 on the line	DO Go straight
ELSE		DO null statement (hold mode)

The above is the logic of “simple line tracing”, which starts from two sides to the middle to adjust GSS status; then the line losing can be highly avoided. Besides, the robot can adjust the GSS status from the middle to its two sides, which also can improve the speed of line tracing.

Figure D



The following is the program logic of “Line Tracing by Intersection”.

Line Tracing by Left Intersection: (End: Crossing the intersection).

```
WHILE   GSS0 off the line     DO    simple line tracing
WHILE   GSS0 on the line      DO    null statement (hold mode)
```

Line Tracing by Right Intersection:

```
WHILE   GSS4 off the line     DO    simple line tracing
WHILE   GSS4 on the line      DO    null statement (hold mode)
```

We can realize the travelling path in the figure D by the following statement:

Take left intersection as the sign:

```
FOR     3 times loop   DO    line tracing by left intersection
End
```

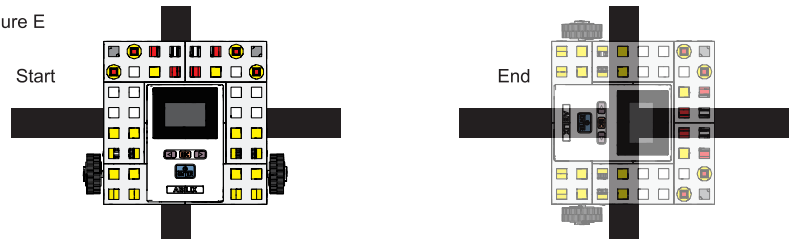
Take right intersection as the sign:

```
FOR     2 times loop   DO  line tracing by right intersection
End
```

The robot needs turning during the line tracing. Figure E shows “turn right”, which is still controlled by the GSS: the middle GSS (2) moves along the current black line to the target black line. As for the GSS2, its initial status is on the line. Therefore, we can’t simply put the status of the GSS as the sign of start & end.

We take the right turn for example to analyze the status of the GSS. It can be concluded: the target black line will scan through GSS4, 3, 2 and stop at GSS2. As mentioned in the “line tracing by intersections”, the GSS used for searching for intersections will cross the black line instead of right on the line, e.g. when “Line tracing by right intersection” stops, the GSS4 won’t be on the black line; then the status of the GSS can be adjusted when the robot turns right; once the GSS is on the line, the GSS3 status can be adjusted…till the GSS2 is on the line.

Figure E



From above, the program logic of “Turn Right” can be concluded as:

```
WHILE   GSS4 off the line   DO   turn right
WHILE   GSS3 off the line   DO   turn right
WHILE   GSS2 off the line   DO   turn right
End
```

Likewise, the program logic of “Turn Left” can be written as:

```
WHILE   GSS0 off the line    DO    turn left
WHILE   GSS1 off the line    DO    turn left
WHILE   GSS2 off the line    DO    turn left
End
```

Conclusion: Generally speaking, the robot traces line by intersections before turning. It traces the line of the direction where it needs to turn to.

After solving two problems—“Line Tracing by Intersection” and “Turning”, most routes in the arena can be achieved. What about there is no intersection for positioning? As for the condition shown in the figure F, we can adopt “Line Tracing by Time” to solve this problem, the program logic is listed as below:

Start Timing

```
WHILE  unfinished time   DO     simple line tracing
End
```

Figure F

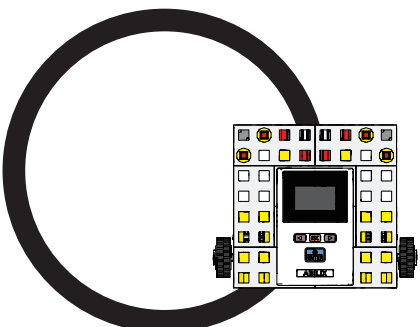
Under the same condition as in figure F, if there is an obstacle at the end, we can detect the obstacle with IR sensor, with one condition set as: the robot stops when the IR sensor detects any obstacle. The program logic is shown as below:

```
WHILE    condition not met    DO    simple line tracing
End
```

By this point, we have known all the principles used in line tracing. Next, we will discuss robot's moving posture.

As a two-wheel differential driven robot, its moving posture is decided by two wheels' speed. In line tracing, the optimal status is that the robot can move as the corresponding line directly under setting values, while robot can just move a similar route in actual programming.

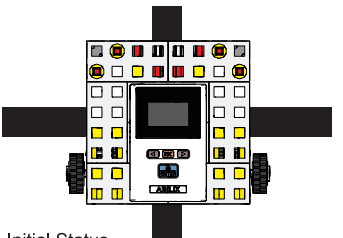
Figure G



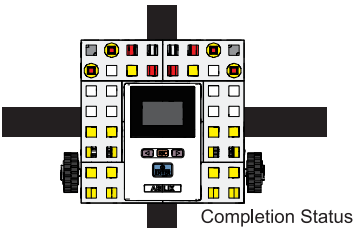
As shown in figure G, the robot needs to “draw” a circle; we can adjust the speed value of “Go Straight” statement in the “Simple Line Tracing” program. The robot in the figure G needs to move counterclockwise; the speed of left wheel L should be slower than that of right one R. However, the actual difference value is an empirical value, which can be obtained in iterative debugging.

Likewise, speed value should be taken into full account in setting “Turning”. As shown in figure H, “Completion Status A” can be achieved by rotating clockwise, with the right wheel as the center (R=0); if two wheels’ speed is set as “L=50, R=50”, and then you will see the “Completion Status B”, which rotates on the center of the line between two wheels.

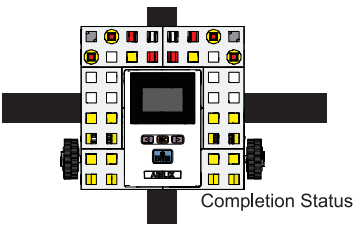
Figure H



Initial Status



Completion Status A



Completion Status B

3.Program Design of Some Common Routes

3.1 Setting of “Initialize” block

Each program that calls blocks of “Line Following blocks” should start from this “Initialize” block.

Motor: the option should be unchecked since the closed-loop motor isn’t included in C203.

Port: used for confirming the port of motor and GSS. Parameters are configured according to actual wiring.

Power: It’s a coefficient. All the speed parameters in the “Line Following Blocks” should multiply by this coefficient. If the robot moves backward when some motor rotates clockwise, the coefficient can be set as negative.

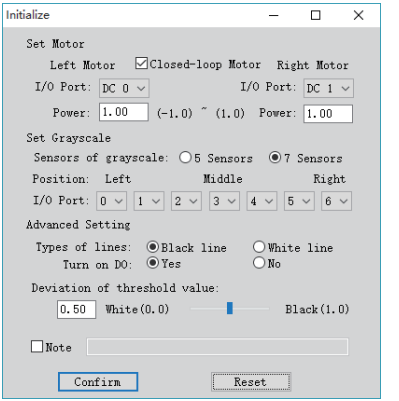
Turn on DO: “Yes” is suggested. In this case, the sensor is power on when the program starts to run.

Deviation of threshold value: threshold value is generally assumed by:

Threshold value: (White value + Black value) × 0.5

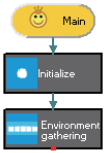
The blacker it is, the greater the value will be.

If it’s easy for the robot to lose lines (can’t find black lines), “0.5” can be reduced; if it’s too easy for the robot to recognize lines (recognize the black ground as black line), “0.5” can be added.



3.2 The usage of “Environment Gathering” block

This block is only used when gathering the threshold value of GSS is needed, with sample program shown on the right. When it’s downloaded to the controller, users can set according to the interface. This block, which is not a required block in the program, is not suggested to put in each program since there will be more button operations. It’s suggested to download this block separately to the controller, and run it when it’s necessary.



3.3 Line Following Block

Type of Intersection: it depends on the direction of the next movement. If it’s not turning, the side with fewer intersections is generally chosen.

Speed Setting: the relation of two wheels’ speed value is as below:

Speed of left wheel

=line tracing speed-speed difference of left turn

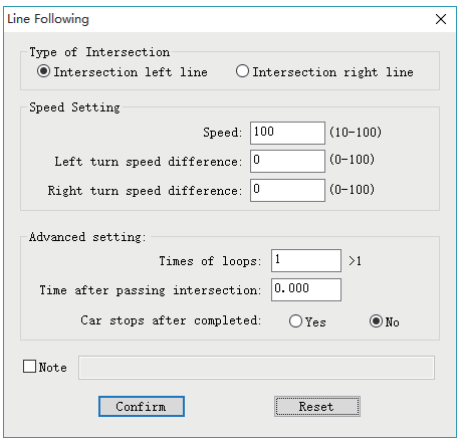
Speed of right wheel

=line tracing speed-speed difference of right turn

This function is used for arc running. Speed difference of right turn is required to set when the robot is running along arc clockwise.

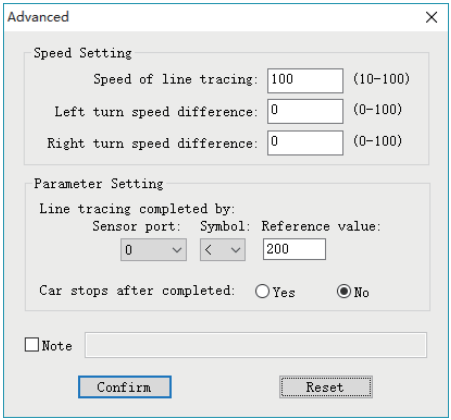
Segmental debugging is suggested to adopt during the debugging. In the final block, “robot stops after completion” should be selected.

It’s also suggested to write some necessary instructions in the “Note” for some key blocks so that it’s easy to read and modify.



3.4 “Advanced” Block

This is a “sign” for the end of line following, namely this block stops when this condition is met. It’s generally used for positioning when there is no intersection sign, or finding line again after losing it.



3.5 Processing Scheme of Various Lines

A. Line Following at Special-Shaped Intersection

As shown in the figure I, no matter what shape or angle of the extended black line is, it’s an intersection in the line following. If it’s an intersection with obtuse angle (i.e. it’s not the intersection that is firstly detected by the outmost GSS), the speed difference of the motor on the other side should be set so that the intersection can be detected earlier. Besides, “Time after passing intersection” can be added to solve the interference of special-shaped intersection.

B. Turning at Special-Shaped Intersection

If the robot needs to turn at a special-shaped intersection, it must trace the line on the target turning side to guarantee GSS of that direction is on the right place.

If the lines around the turning are very dense, as shown in the figure J, the robot can move off the line under permitting rules, namely when the robot enters the first intersection of the circle, “time after passing intersection” can be set a little bit longer for the left wheel to arrive at area A. After that, the turning motion can be set as crossing two intersections on the left wheel (L=0); then the robot moves forward along the line again.

Figure I

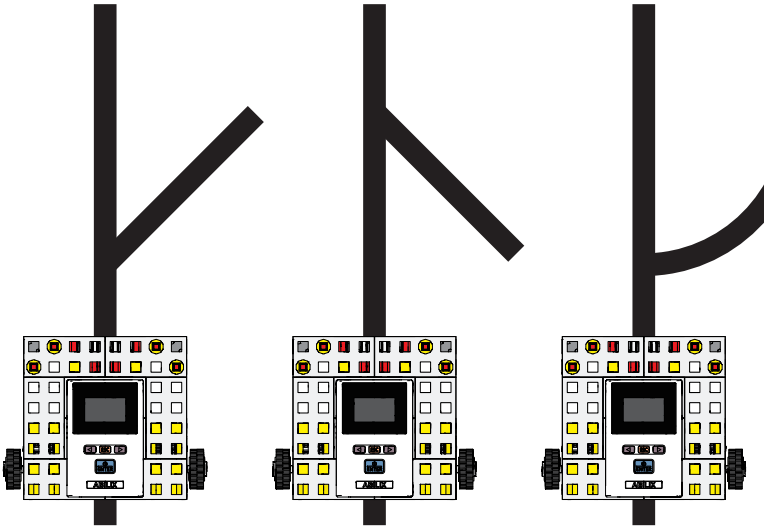
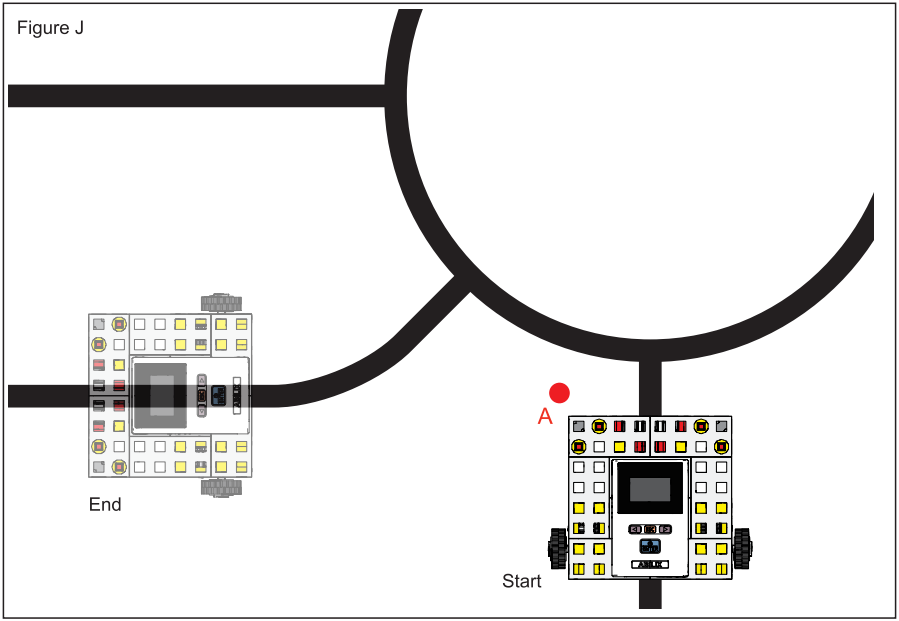


Figure J



C. Offline Moving

As shown in the figure K, if the robot moves by the book, there will be many troubles. In this case, offline moving can be adopted, of course, if rules allow.

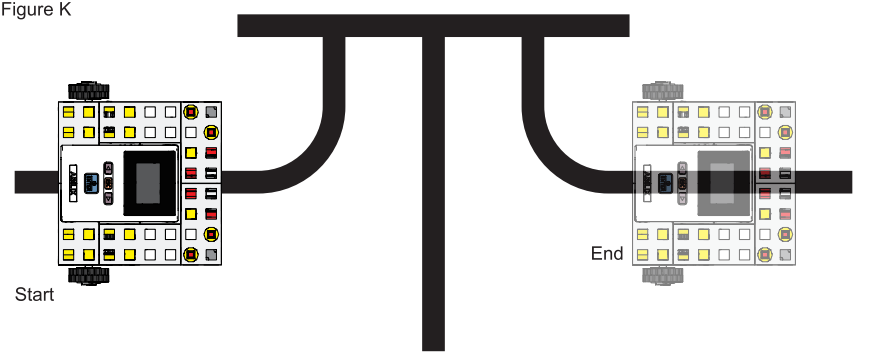
① Modify the program to make sure the robot's direction in the "Start" is the same with the target one.

② Select the "Sensor" in the "Start Motor" so that the robot can find the vertical black line on its way forward.

③ Use the "Delay" function in the "Start Motor" to make the robot move forward for a short distance. Try to make the robot as close to the end as possible while not pass it, by this time the GSS has found the black line.

④ The robot arrives at the end by line following function.

Figure K



4. Suggestion for Debugging

Task models will be placed differently in the actual competition, with other extra tasks. Under such background, the debugging is suggested to be participants-oriented and coach-guided.

4.1. Comprehensive Consideration

The robot starts from the base and returns to the base; this process is called a line. Each route is optimized by players' ability, time saving and stability.

a. Finish more tasks along one line. (Time Saving)

b. The longer the line is, the more difficult the programming will be. (Programming Ability Testing)

c. The longer the route is, the higher error rate will be. (Lower Stability)

Versatility and stability also should be considered in the mechanism design.

Finish more tasks with one mechanism.

Guarantee the stability of task completion

Choice

a. Give priority to the tasks with high scores.

b. Guarantee tasks with high success rate; give up lower ones.

4.2. Design a Route

The route is suggested to be designed as the following steps:

a. Overall Strategy: consider the task number from mechanism function.

b. Route Design: how to traverse all the tasks point more efficiently.

c. Mechanism Design: refine mechanism scheme of the "Overall Strategy".

d. Mechanism Building and Testing: basic requirement: firm and concise mechanism.

e. Program Design: ①Route Debugging ②Motion Debugging ③Joint Debugging ④Success Rate Verification

f. Task Choice: remove tasks that are hard to realize from the line; other tasks also can be strung to this line and finished.

4.3. Game Simulation

After all lines are finished, have a simulation competition. Each team is divided into judge team and participants, with scores and time formally recorded. In this case, violation action in the strategy can be discovered. On the other hand, something ambiguous in the rules can be found.

5. FAQ & Solution

Robot speed can't be controlled by the parameters.

Uncheck the "Closed-loop Motor" in the "Initialize" block.

Robot loses the line.

Critical Value Problem: have environment collected again

Hardware Problem: Detect connecting condition of ports and GSS property on the "AI" interface. Detect connecting condition of ports and GSS property on the "Motor" interface.

Program Problem: Find the corresponding block according to errors, and modify it.

6.Bill of Materials


Cube
*14


Half cube
*6


Slope cube(45°)
*2


Shaft Sleeve
*31


Beam(2)
*5


Beam(3)
*18


Beam(7)
*23


Beam(11)
*9

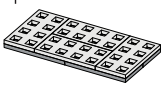

Beam(3*5)
*18

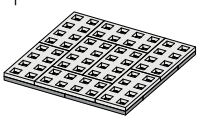

Beam(4*6)
*6


Slab(1#)
*8


Slab(2#)
*3


Slab(3#)
*1


Slab(4#)
*1


Slab(8#)
*1


Cube connector
*8


Pin(20)
*46


Pin(30)
*27

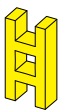

Motor Connector
*1


Axle Connector(180°)
*5


Axle Connector(90°)
*5


Short bolt
*8


L Connector
*18


H Connector
*3


A Connector
*5


Gear(12)
*9


Gear(20)
*3

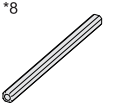

Gear(28)
*8


Gear(12/28)
*6


Axle(20)
*2


Axle(40)
*8

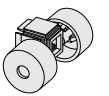

Axle(60)
*8


Axle(80)
*10

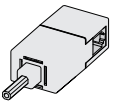

Hub
*2


Tire
*2


Bearing
*10


Guide Wheel
*1


Grayscale Sensor
*5


Motor
*3


Controller
*1

Motor Wire*3

Data Cable*1

User's Guide for Ccon102*1

CD*1

Note: Battery pack (adaptor or lithium battery) is not included in this kit. Users need to prepare at least 6 unused AA batteries. As there are many electron components and battery types mentioned in the user's guide of controller, please filter the content before study.

WER
World Educational Robot Contest

www.wercontest.org

Shanghai xPartner Robotics Co., Ltd.
8th Floor,Building 90,No.1122
North Qinzhou Rd
Shanghai,China,200233
www.abilix.com