Dragit

Profile of Dragit

Install Dragit

Before any operations, you need to log into the Raspberry Pi remotely first. Here's how:

1 · Get the IP address of the Raspberry Pi:

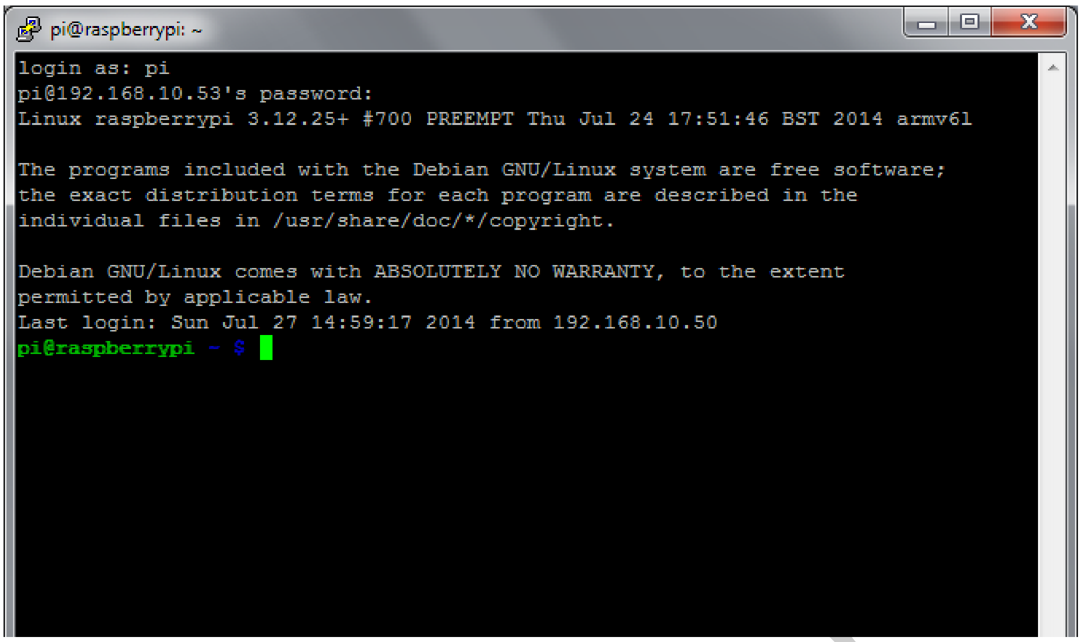
https://www.raspberrypi.org/documentation/remote-access/ip-address.md

2 · Log in with the IP address, by ssh or VNC:

For ssh: https://www.raspberrypi.org/documentation/remote-access/ssh/README.md

For VNC: https://www.raspberrypi.org/documentation/remote-access/vnc/README.md

Now you're on the RPi already. Open a terminal for command lines. Let's see what next.



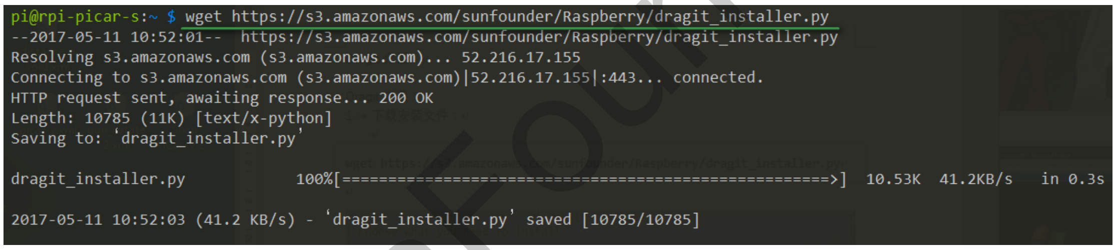
```
login as: pi
pi@192.168.10.53's password:
Linux raspberrypi 3.12.25+ #700 PREEMPT Thu Jul 24 17:51:46 BST 2014 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Jul 27 14:59:17 2014 from 192.168.10.50
pi@raspberrypi ~ $
```

1. Type in the command below:

```
wget https://s3.amazonaws.com/sunfounder/Raspberry/dragit_installer.py
```



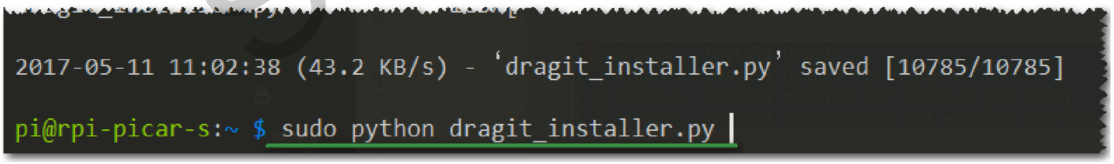
```
pi@rpi-picar-s:~ $ wget https://s3.amazonaws.com/sunfounder/Raspberry/dragit_installer.py
--2017-05-11 10:52:01--  https://s3.amazonaws.com/sunfounder/Raspberry/dragit_installer.py
Resolving s3.amazonaws.com (s3.amazonaws.com)... 52.216.17.155
Connecting to s3.amazonaws.com (s3.amazonaws.com)|52.216.17.155|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10785 (11K) [text/x-python]
Saving to: 'dragit_installer.py'

dragit_installer.py          100%[=====================================================>]  10.53K  41.2KB/s   in 0.3s

2017-05-11 10:52:03 (41.2 KB/s) - 'dragit_installer.py' saved [10785/10785]
```

2. After download is done, type in the command to run:

```
sudo python dragit_installer.py
```

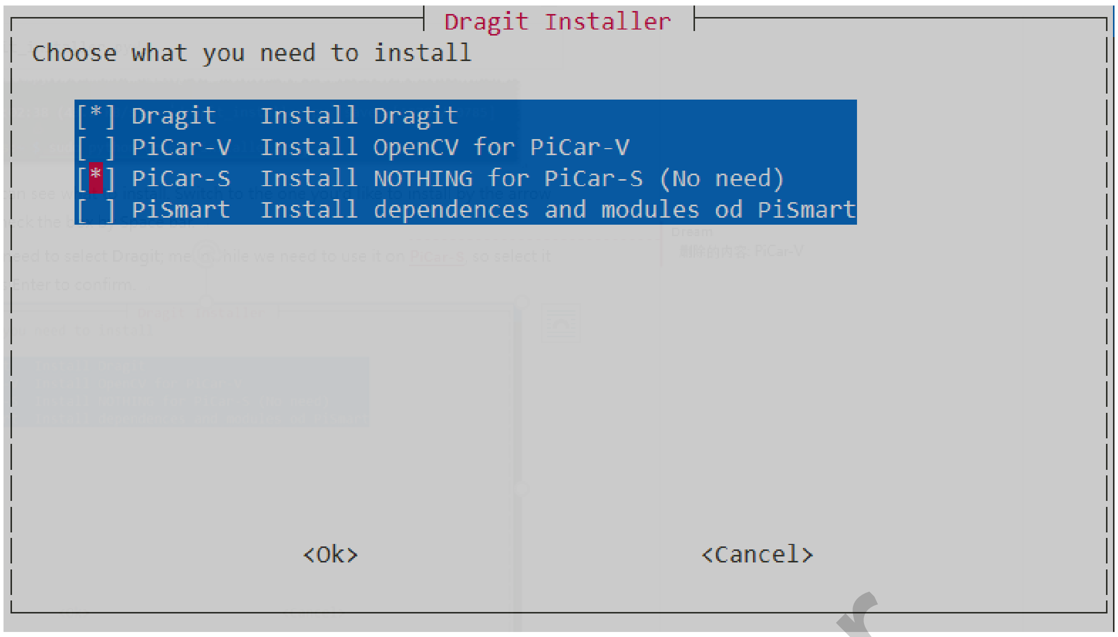


```
2017-05-11 11:02:38 (43.2 KB/s) - 'dragit_installer.py' saved [10785/10785]

pi@rpi-picar-s:~ $ sudo python dragit_installer.py
```

3. Then you can see what to install. Switch to the one you'd like to install by the arrow key, and check the box by Space bar.

Of course you need to select **Dragit**; meanwhile we need to use it on **PiCar-S**, so select it too. Then press Enter to confirm.

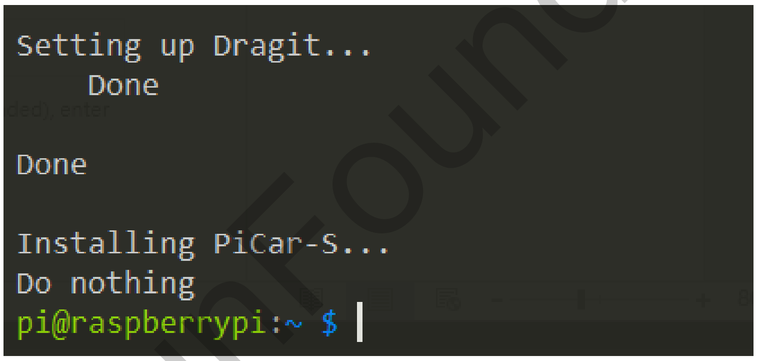


```
Dragit Installer
Choose what you need to install

[*] Dragit    Install Dragit
[ ] PiCar-V   Install OpenCV for PiCar-V
[*] PiCar-S   Install NOTHING for PiCar-S (No need)
[ ] PiSmart   Install dependences and modules od PiSmart

          <Ok>                      <Cancel>
```

Then the program will process the installation. You need to wait for a while, and pay attention not to disconnect the network during the process.



```
Setting up Dragit...
    Done

Done

Installing PiCar-S...
Do nothing
pi@raspberrypi:~ $
```

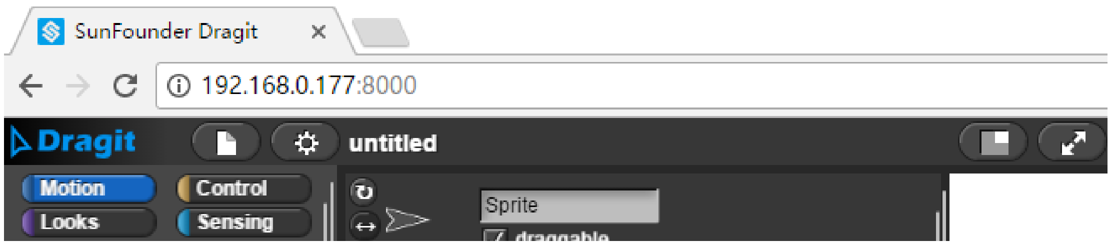
When the prompt appears as shown above, it means the installation is done.

After installation, reboot:

```
sudo reboot
```

On your computer/tablet, open a web browser (Chrome/Firefox/Safari recommended), enter the PI address of the PiCar-S and then 8000:

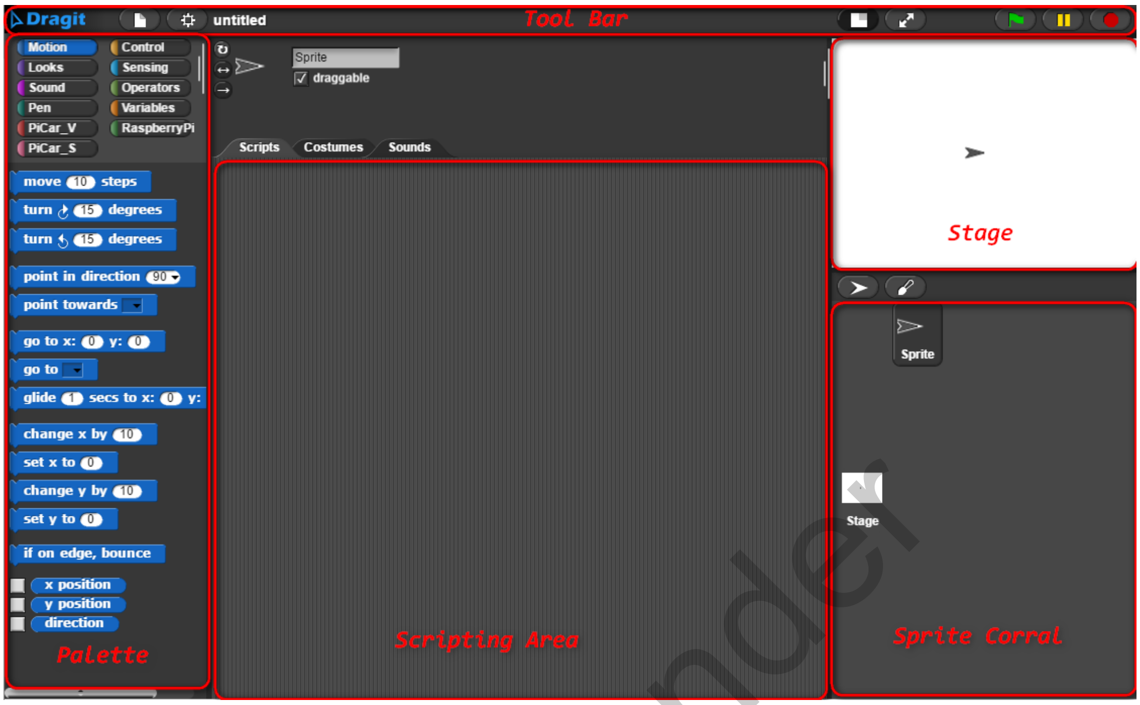
E.g. 192.168.0.177:8000



Then you're in the Dragit window.

Use Dragit

After opening Dragit, you can see the following window:



There can be one or more scripts for a project in Dragit. The scripts are shown in the scripting area and constructed by blocks in the palette.

In this tutorial for how to use Dragit for PiCar-S, we'll focus on the category of PiCar-S, basics of using Dragit, and how to control PiCar-S in this software.

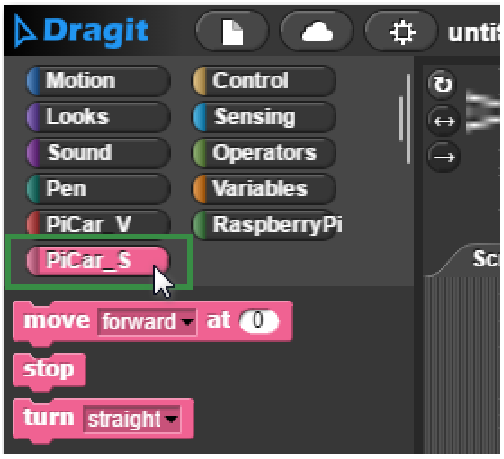
Lesson 1: Move Your Car Forward

The basic movements of the PiCar-S, just like any other car robots, are to go forward and backward. So let's start learning from its moving ahead and back.

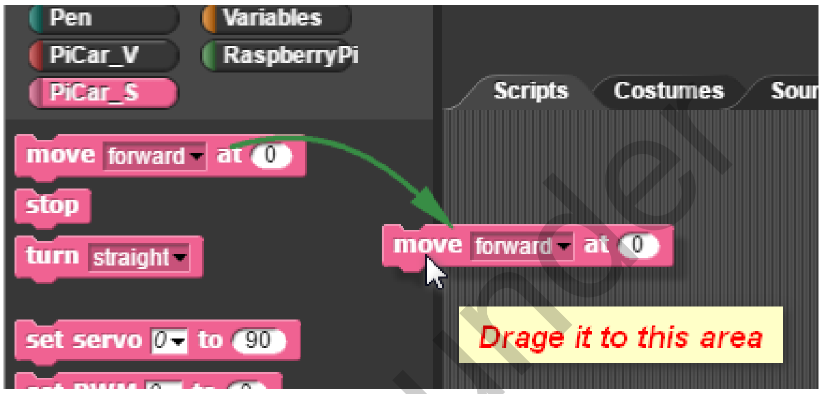
Before operation, in case the car runs out of the desk and falls off, we prop it off the surface.

And then take these steps:

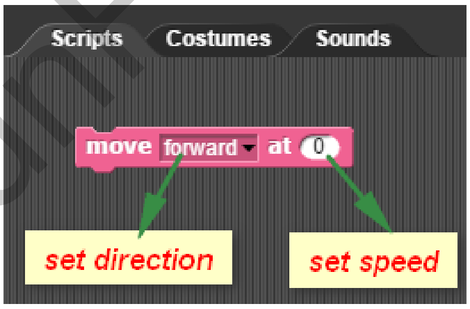
1 · In the **palette** area, click to select the PiCar_S category.



2 · Then click and drag move forward at 0 into the **scripting** area.



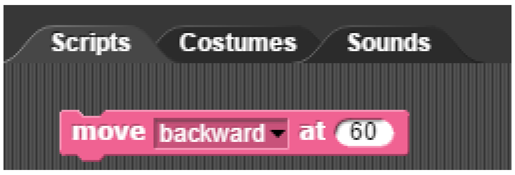
3 · So now in the scripting area there's already a block to control the robot to run forward and backward. Let's check how it is done with the block!



In this block, there are two parameters for control: direction and speed.

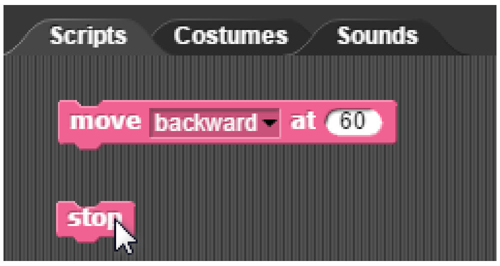
Click the input of direction by the **triangle** sign and select a value at the drop-down list, and then click the speed and enter a number ranging between 0 and 100. Try 60 for example. So now your block includes the speed and direction which control the car to move.

Click this block and see what will happen to the car. Yes! The car's back wheels start to spin and it will run on the table if you take it down.



Click the direction input and select backward, click the block again, and the back wheels will spin reversely, so the car will go backward if you put it on the table.

OK, now the robot may get tired after running and you can get it to stop. Click and drag stop to the scripting area and click the block. Good, have a good rest.



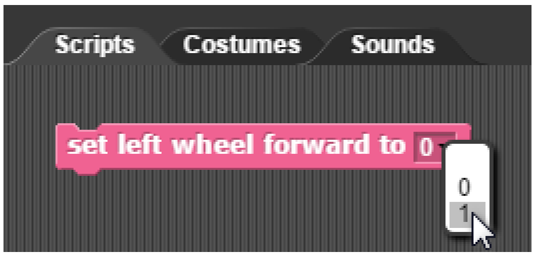
Special Notes

Generally your PiCar-S will move as you control. But, something will possibly go wrong. For example, when you set it to move forward, it goes backward; input backward, it may go forward. Or, even unexpectedly, the two back wheels spin in different directions…

Chillax! You can calibrate them right with these blocks: set left wheel forward to 0 and set right wheel forward to 0.

If the left back wheel goes the wrong direction, drag set left wheel forward to 0 into the scripting area, change the input, and click the block; if the right wheel is not functioning well, drag set right wheel forward to 0 similarly; just drag them both if two back wheels need to correct.

For example, I need to change my left back wheel, I'll drag set left wheel forward to 0, set input to 1, click the block and the wheel will reverse the spinning. If I happen to dislike the adjustment (though I don't know why LoL), I'll change the number back to 0 and click the block, and the direction will change back again.



So now it's your turn to adjust the spinning direction of your car wheels. Just try!

Lesson 2: Turn Your Car Around

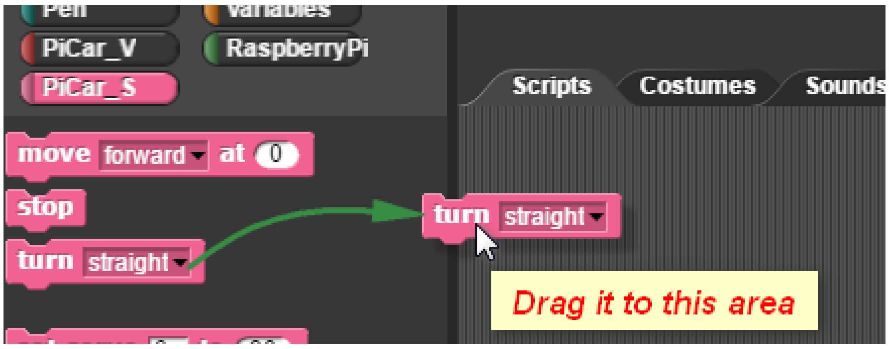
Now after the Lesson 1, you should be able to control the car to go forward and back and make it stop. Then, you may wonder: how can it turn around?

So, next we'll move on to turning of the car.

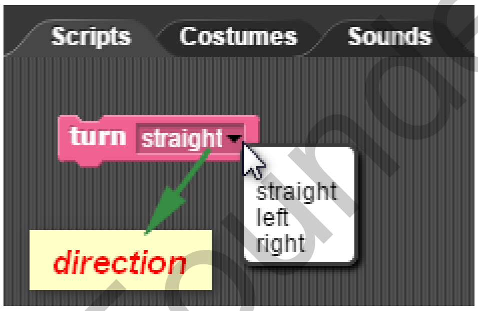
As before, prop the car and leave its wheels off the ground in case of running out of the table. For its turning around, the block turn straight▾ will be used. Very similar steps as in Lesson 1.

1 · Click PiCar_S in the palette area. Or you can just stay there if you're already there.

2 · Click and drag turn straight▾ into the scripting area.



3 · Now in the scripting area there's a block to control the car to turn. Let's check its details.



Click the triangle sign in the block and you can see the values on the drop-down list. Just pick an input to make the car turn in that direction.

For example, select left. Click the block and observe the car.

Yes, the car's front wheels will turn left accordingly, well, because we just made them do.

In the same way, change the direction to straight and right and click the block to run. Then, how are the front wheels going? Pretty easy, right?

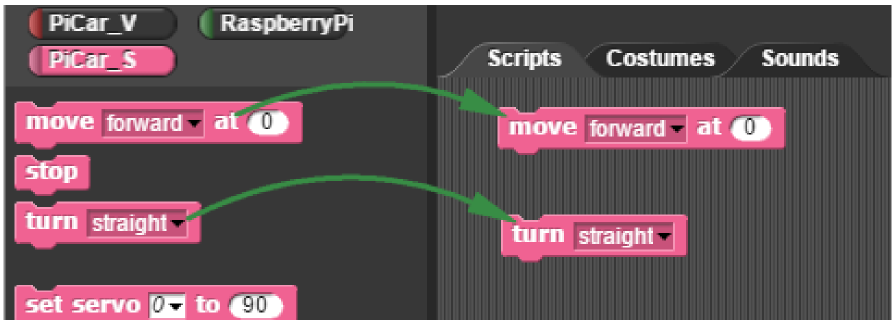
Hm, that may not be what you really want because the wheels just keep turning. You want the car to turn a direction when it's running forward or backward? Of course that's easy to make! No need to rush and we're just about to talk about making it by combining two or more blocks.

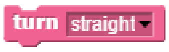
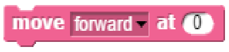
✧ Combined Blocks

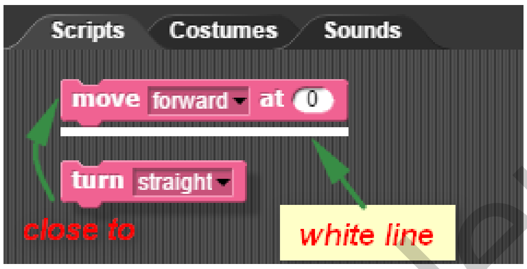
To make the car turn left or right when it's running, we need the car to go forward/backward + turn left/right/straight... which means to make the blocks in Lesson 1 and 2 work at the same time.

So, how to combine these two?

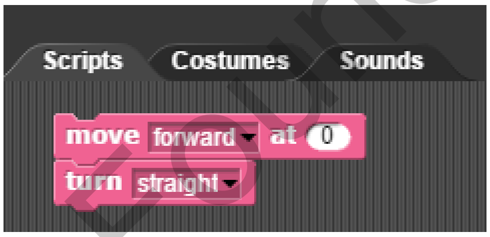
1 · Drag move forward▾ at 0 and turn straight▾ blocks into the scripting area.



2 · Drag  under , approaching until a white line appears between them.

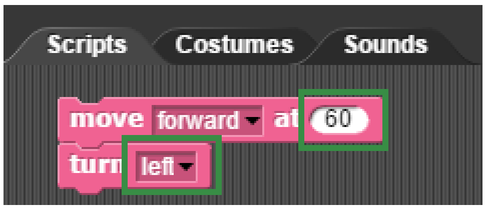


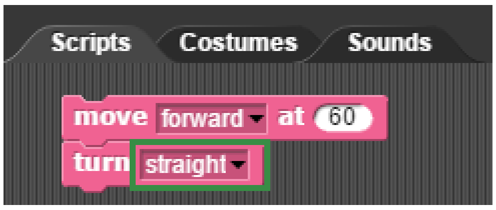
3 · Release the block and you'll see the two are combined!



To run the combined block, just as to do a single one, simply click it - you can take it as a whole, just a bigger block.

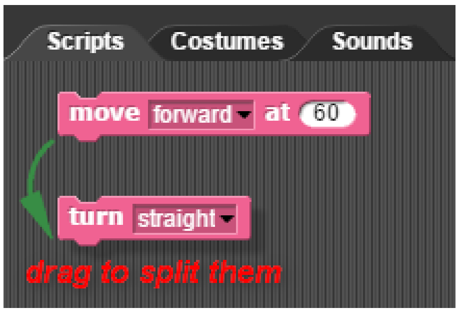
4 · Try to change the inputs in this larger block, click it, and see what happens. As shown in the figure below, as set to "**move forward**" at "**60**" speed and "**turn left**", so car is running slightly fast when turning left.





Or, change "**left**" to "**straight**" and click the block again. Then the car will go straight.

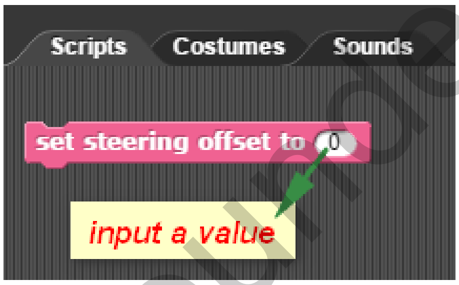
Things can be better if you have other ideas. If I do not feel like combining them, I can just split the two blocks by dragging the lower one to somewhere else, like this:



Special Notes

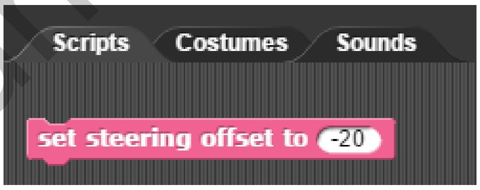
When the car turns "straight", the front wheels are almost oriented to the front. But some minor errors may exist there which can be neglected. However, if it obviously deviates from the right track, you can adjust by set steering offset to 0.

1 · Drag set steering offset to 0 into the scripting area.



2 · Input a value in the block.

If the front wheels deviate to the left, input a positive value, like 20; if to the right, give a negative one like -20 as shown below.



3 · Then click the block and check the front wheels. Still deviations? Try to increase or decrease the value then, based on whether the angle is reduced or over-adjusted compared to the previous status. Just keep calibrating it till you feel it's good.

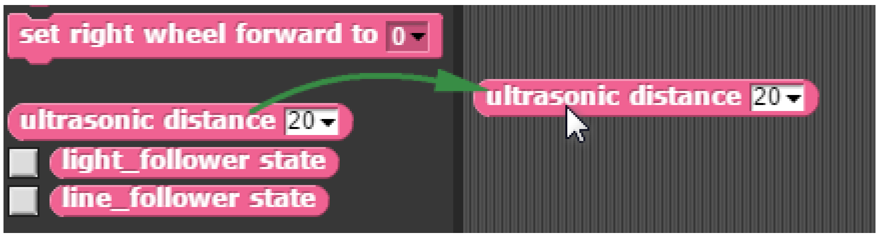
Lesson 3: Get the Ultrasonic Distance

Through Lesson 1 and 2, now I think you can make the car run as you want easily. So those are just basic control of the robot via the Dragit. But definitely that's not all you can learn! To move on, let's check using the software to control the sensors on the car.

Controlling the sensors by the blocks is to let the car act itself, so we don't need to send the commands every time.

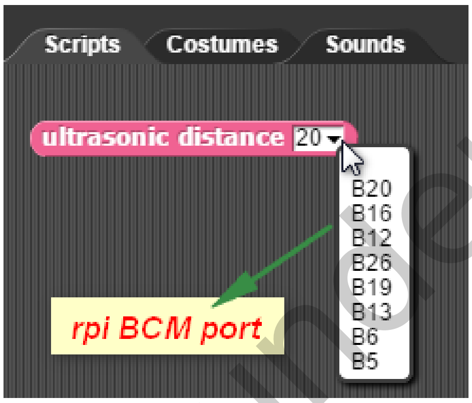
Let's start from the ultrasonic, using this block ultrasonic distance 20.

1 · Drag ultrasonic distance 20 into the scripting area.



2 · Select a port value.

Since the ultrasonic sensor's sig pin connects to B20 in my case, so here I select B20 port. You should select the correct one in your project.

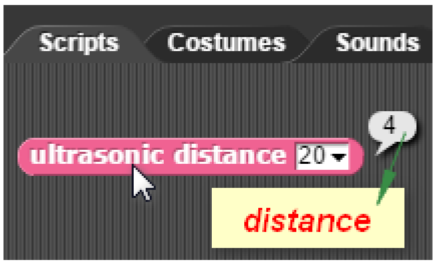


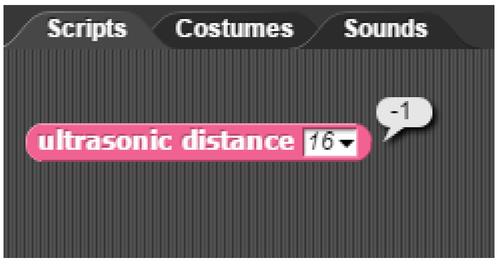
3 · Place an obstacle in near front of the ultrasonic sensor, for example a book, and face the probe of the sensor to the book straight.

Click the block. Next to the block on the right, there will prop a data, which is the distance detected by the ultrasonic sensor, in the unit of cm.

Then move the book (or other gadgets) far away from the sensor or approach it to the sensor, click the block again, and check whether the data popping out changes or not.

If the data returned by the sensor is -1 (in the fig on the right below), it means the detection fails, maybe because the sig of the sensor connects to B20 when port in the block is otherwise.





✧ Store data detected by the sensor

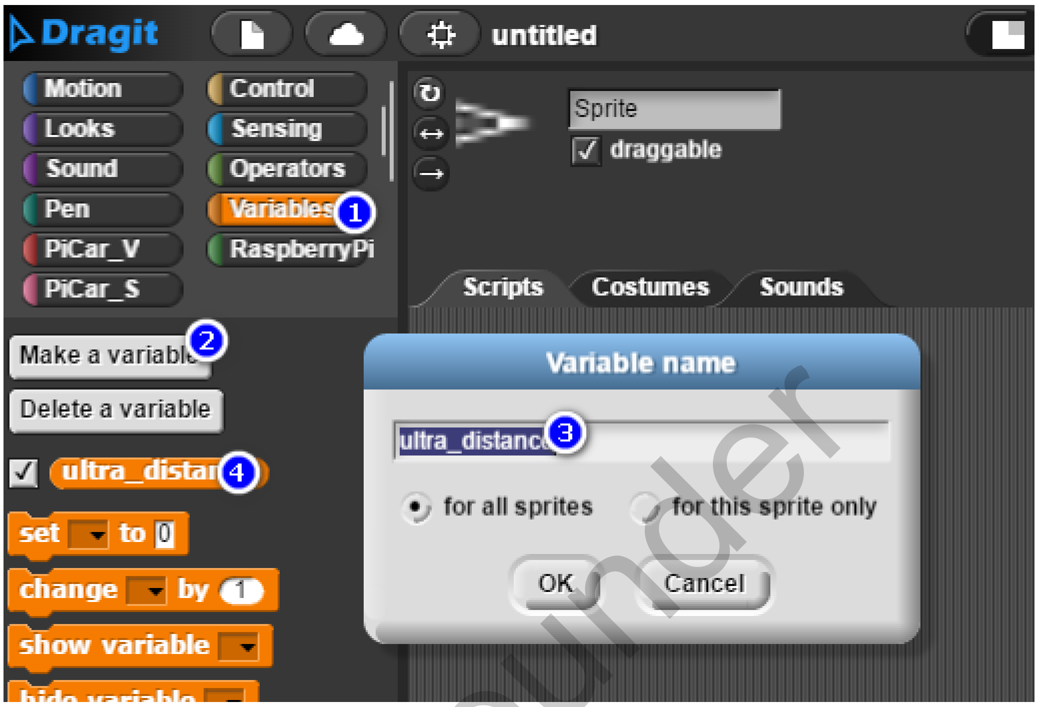
Click the block and you can thus check the data detected by the ultrasonic sensor. To use the data further, we still need to store it. Here we use the block Variables to store the distance.

1 · Select the Variables category in the palette.

2 · Click Make a variable to create a variable.

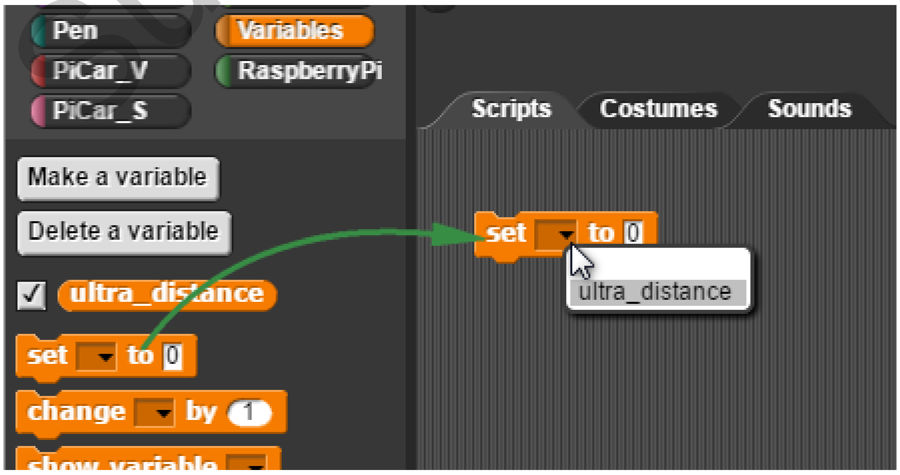
3 · Enter a name of the variable.

4 · Click **OK**. Then you can see the variable in the place marked with 4 in the following figure.

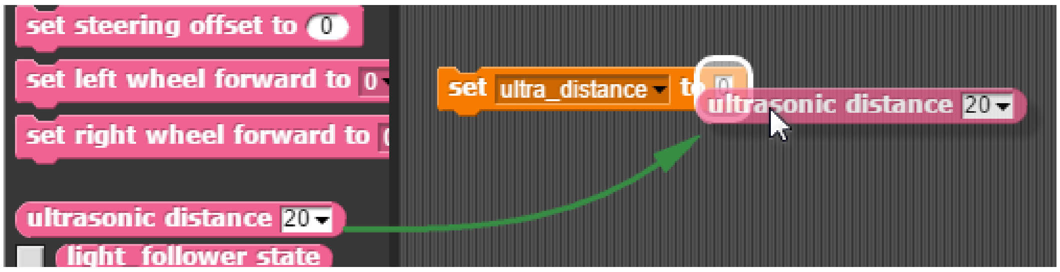


We can take this "**ultra_distance**" variable as a block. Then use the block set to 0 to store the data detected by the sensor in the variable just created.

5 · Drag the set to 0 block into the scripting area, select ultra_distance on the drop-down list.

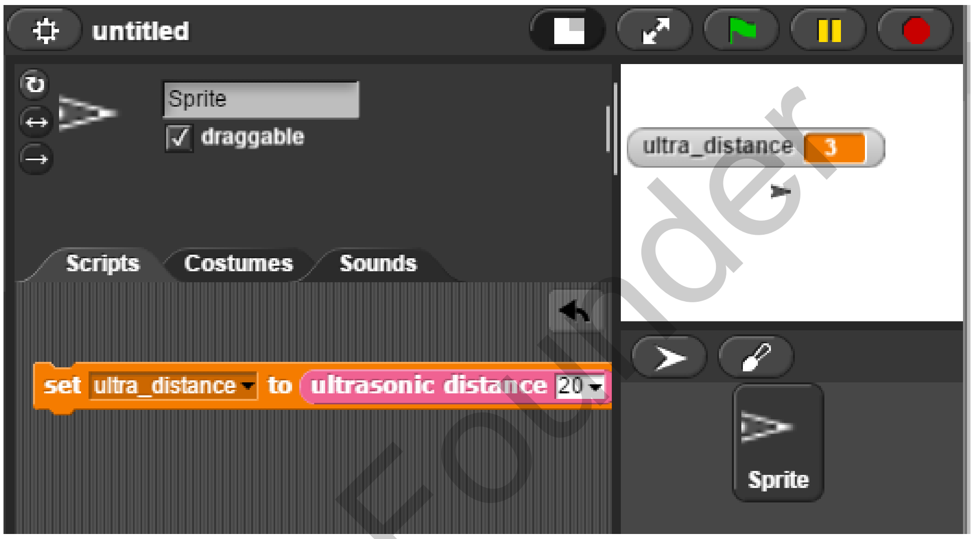


6 · Click the PiCar_S category again, drag the ultrasonic distance 20 to the "playground" (yes, the scripting), to the value of the set to 0 and release. So this block will be taken as the value.



Click this combined block to run, and the data distance detected by the sensor is written to and stored in a variable **ultra_distance**.

In the **Stage** area at the top right corner, you can see the *ultra_distance* variable and its value, the distance detected.



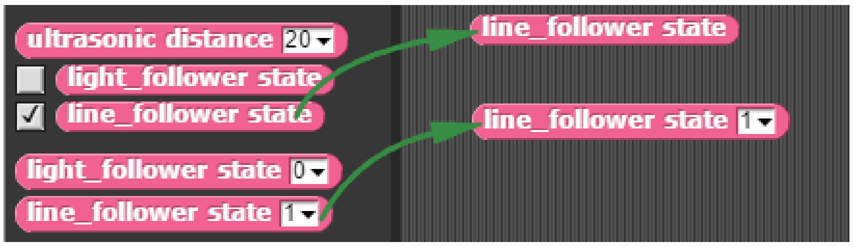
Later we'll see this variable again - we'll use its value later in the automatic control part. Let's move on!

Lesson 4: Get the Light Follower & Line Follower State

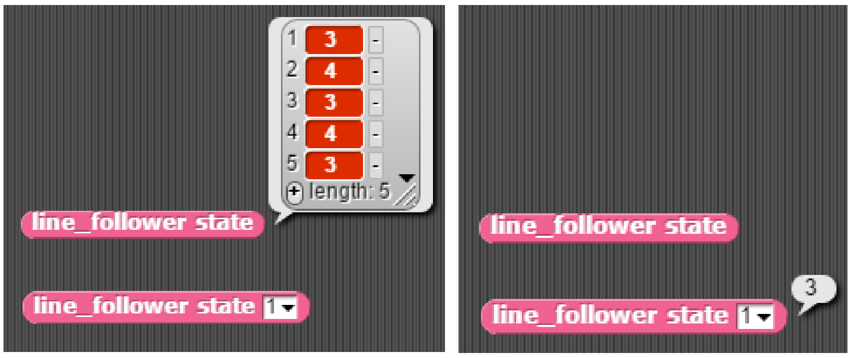
Since these two sensors are similar in application, here we just describe once.

Connect the Light Follower and Line Follower sensors with the car via the cable in the package - hook up the Line Follower and any I2C port, and Light Follower to pin A0, A1, and A2.

1、In the blocks on the palette, find line_follower state, drag them into the scripting area.



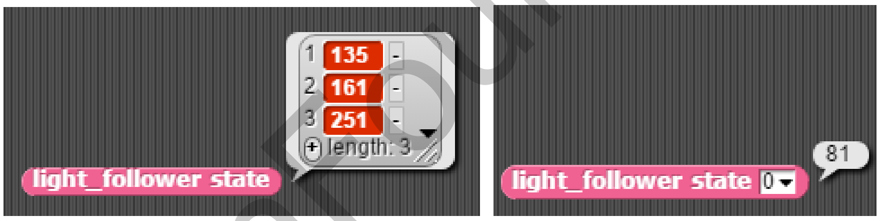
2、Click the two blocks and check the values popped out at the right side.



The value of the line_follower state block is a list of 5 numbers. Hold on, you've seen that there are also 5 probes on the Line Follower sensor? Smart as you are, yes, the values in the list are just the data collected by the 5 probes.

When in line_follower state 1, there's only one value, which is the data detected by one single probe selected on the drop-down list. For example, select 1 on the list, and the data collected by the No. 1 probe will be shown then. Comparing it with that shown on the list previously, you'll find they are amazingly the same! Of course, because they are just the same (LoL).

3 · The Light Follower is similar to the Line one.



The value of the light_follower state block is a list of 3 numbers. Obviously, these 3 values are for the 3 probes on the Light Follower sensor - a single one is mapped to one probe respectively.

When in light_follower state 0, there's only one value, which is the data detected by one single probe selected on the drop-down list. For example, select 0 on the list, and the data collected by the No. 0 probe will be shown then. Comparing it with that shown on the list previously, you'll also find they are just the same!

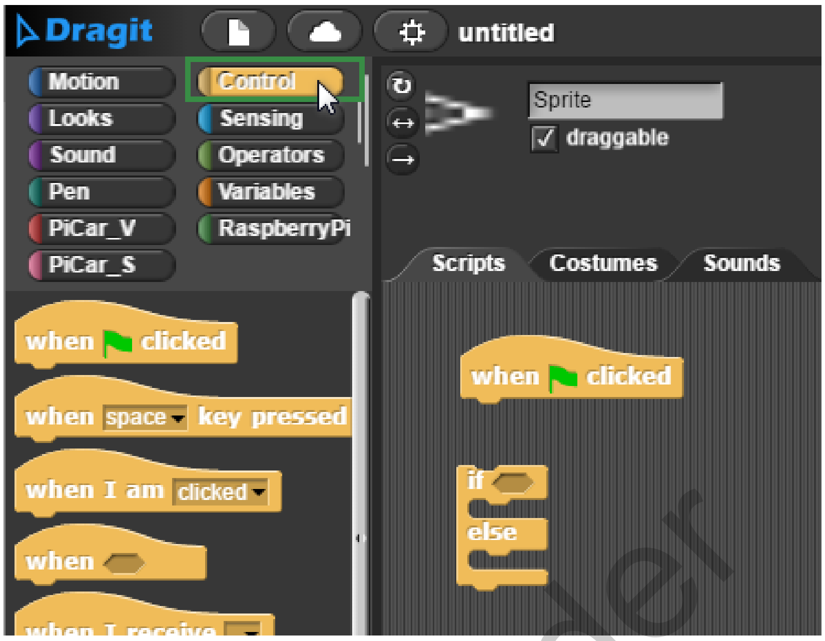
Similar to the ultrasonic sensor, we can create a variable to store the data detected by the sensors. Try by yourself!

Lesson 5: Logic Control

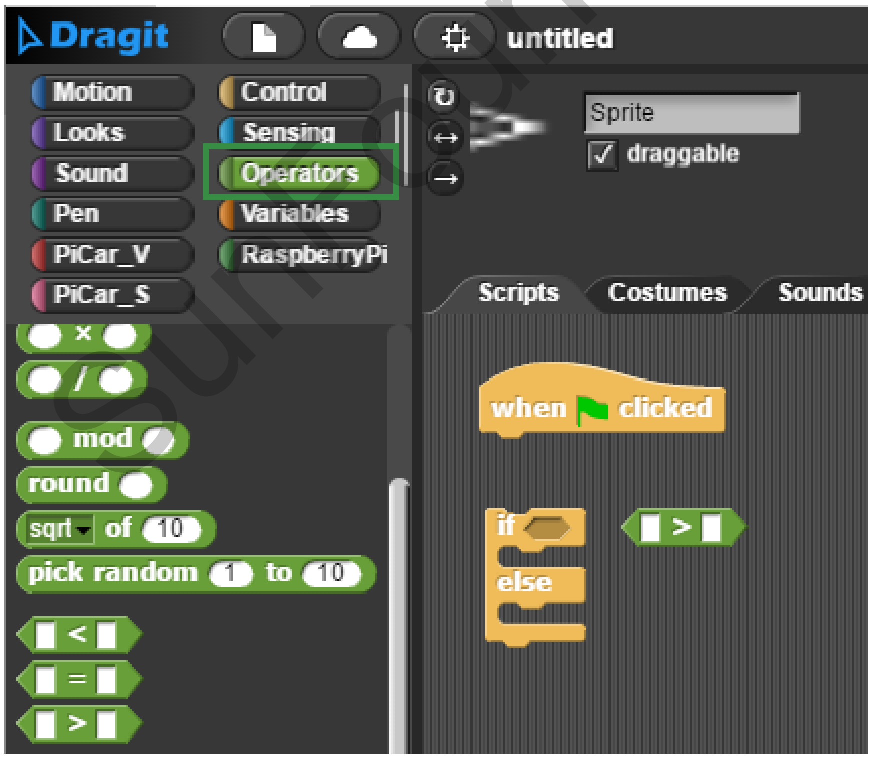
Remember we once mentioned about automatic control for the car? Well, now that you've known how to realize data reading and storage of the sensors, you just need more about logic control to further the control, to the auto mode.

At first there's no program for logic control and the movements are very simple and dull, which on the other hand requires a lot of code work. Our solution is combine different blocks and include logic control for better performance, by using the block Control here. Let's check how to make!

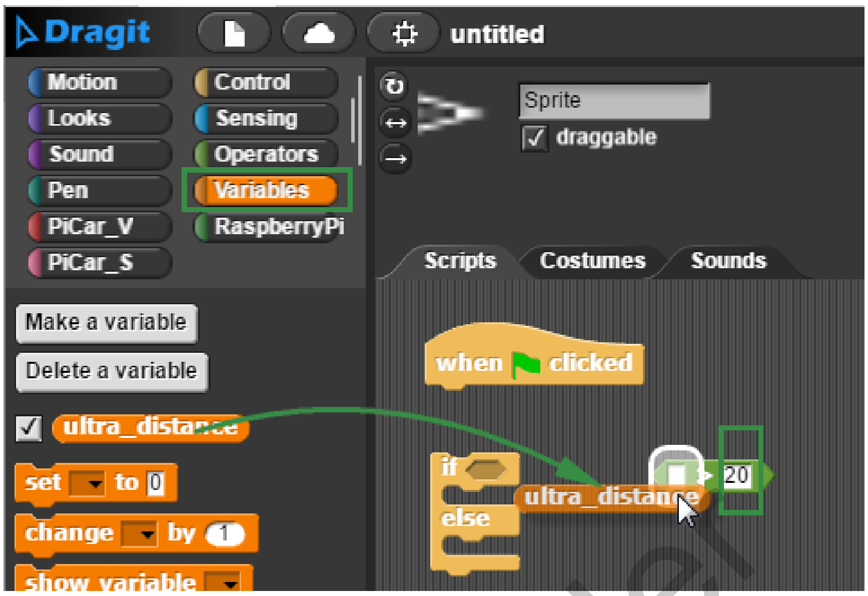
1 · Select Control category in the palette and drag blocks in this category to the scripting area.



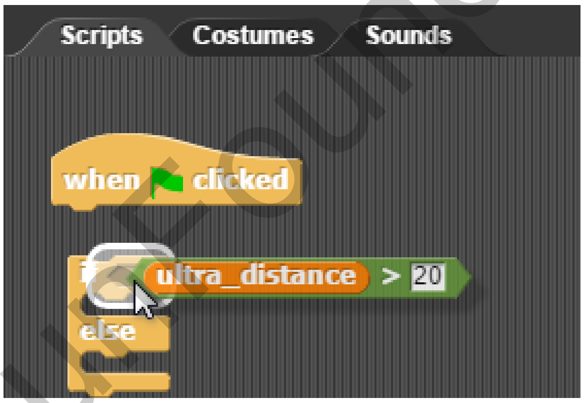
2 · Drag blocks from the Operators category into the scripting area.



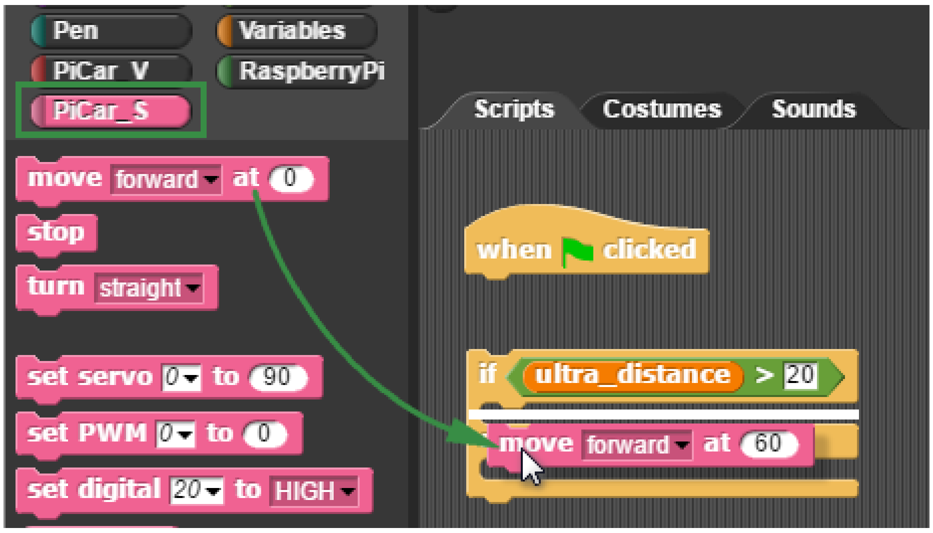
3 · In Variables, drag the created ultra_distance to the left input of the comparison block, and enter a value in the right input, like 20.



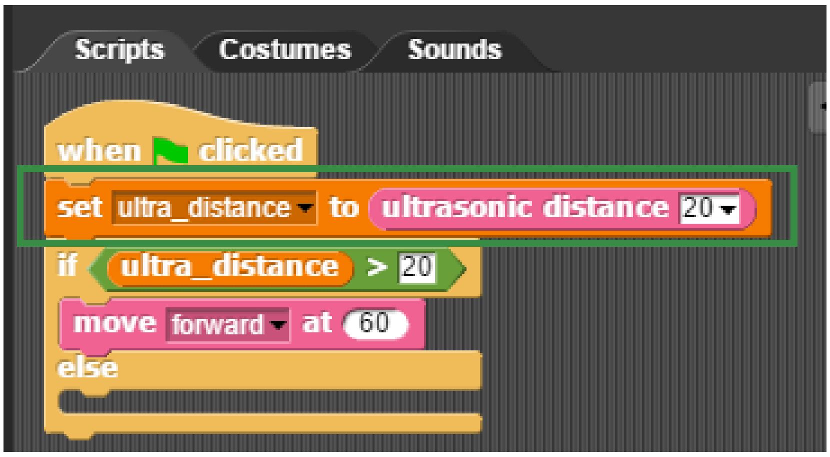
4 · Drag the combined comparison block into the condition input of the if block.



5 · Drag a move forward at 0 block from the PiCar_S category into the input of the if block, and enter a speed value in the input.



6 · Based on Lesson 3, store the data collected by the ultrasonic sensor in ultra_distance block, and combine the blocks, as shown below.

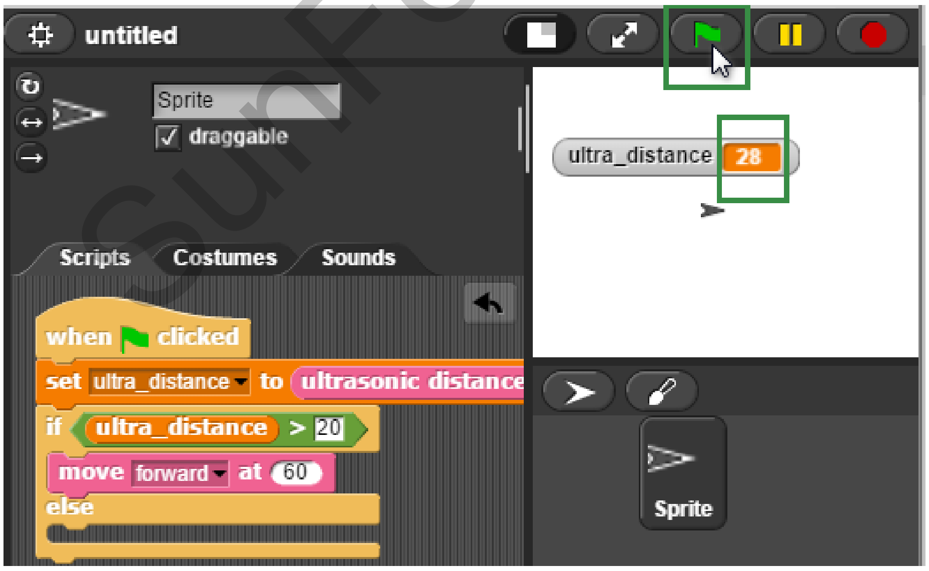


```
when [flag] clicked
set ultra_distance to ultrasonic distance 20
if ultra_distance > 20
    move forward at 60
else
```

7 · So here we have this little combined logic block! So, how does it work? Well, after you run it, when the ultrasonic sensor detects a distance smaller than 20, the car will move forward.

Click the tiny green flag on top to run the combined block again. Observe the value of ultra_distance changes in the Stage area.

In Lesson 3, we used a book as an obstacle and moved it to see the distance change. Here repeat the test again. Move the book away from the sensor and click the flag. Then you can see the distance detected by the ultrasonic increase. When the distance reaches 20, the car will start to move forward.



```
when [flag] clicked
set ultra_distance to ultrasonic distance
if ultra_distance > 20
    move forward at 60
else
```

You make more automatic control with further ideas of complicated logic control. Now try it yourself! Welcome to share with us by post on our FORUM.

Lesson 6: Samples for Automatic Control

Congrats! Arriving here, you have mastered most of the skills of playing PiCar-S with Dragit. Next let's check more samples.

In this lesson, we'll probe into the sample details. Though many blocks are involved, they are rebuilt mostly based on what we've seen in previous lessons. So when you're not sure how the program goes, just look back on the parts like movement control, variable creation, sensor data acquisition, logic control, etc.

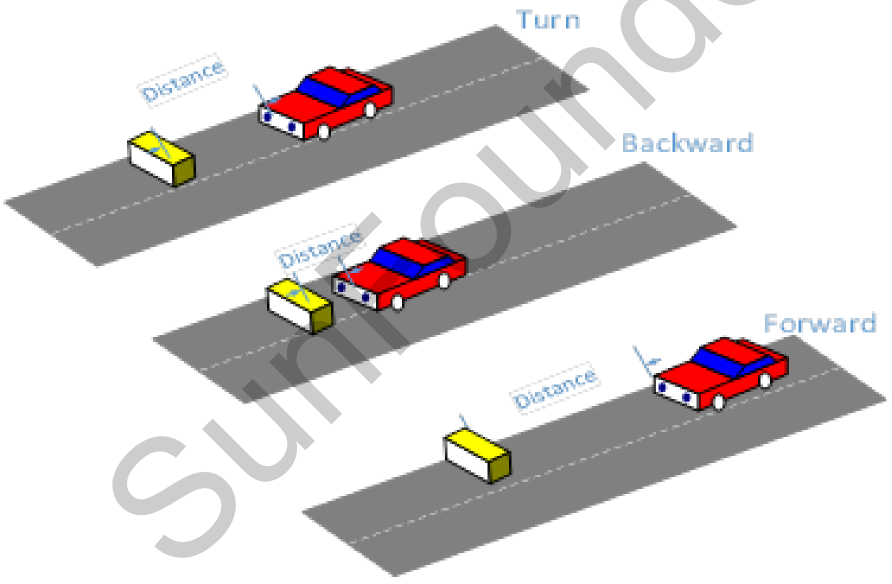
After all these accumulations, we are now ready to sail. Let's rock!

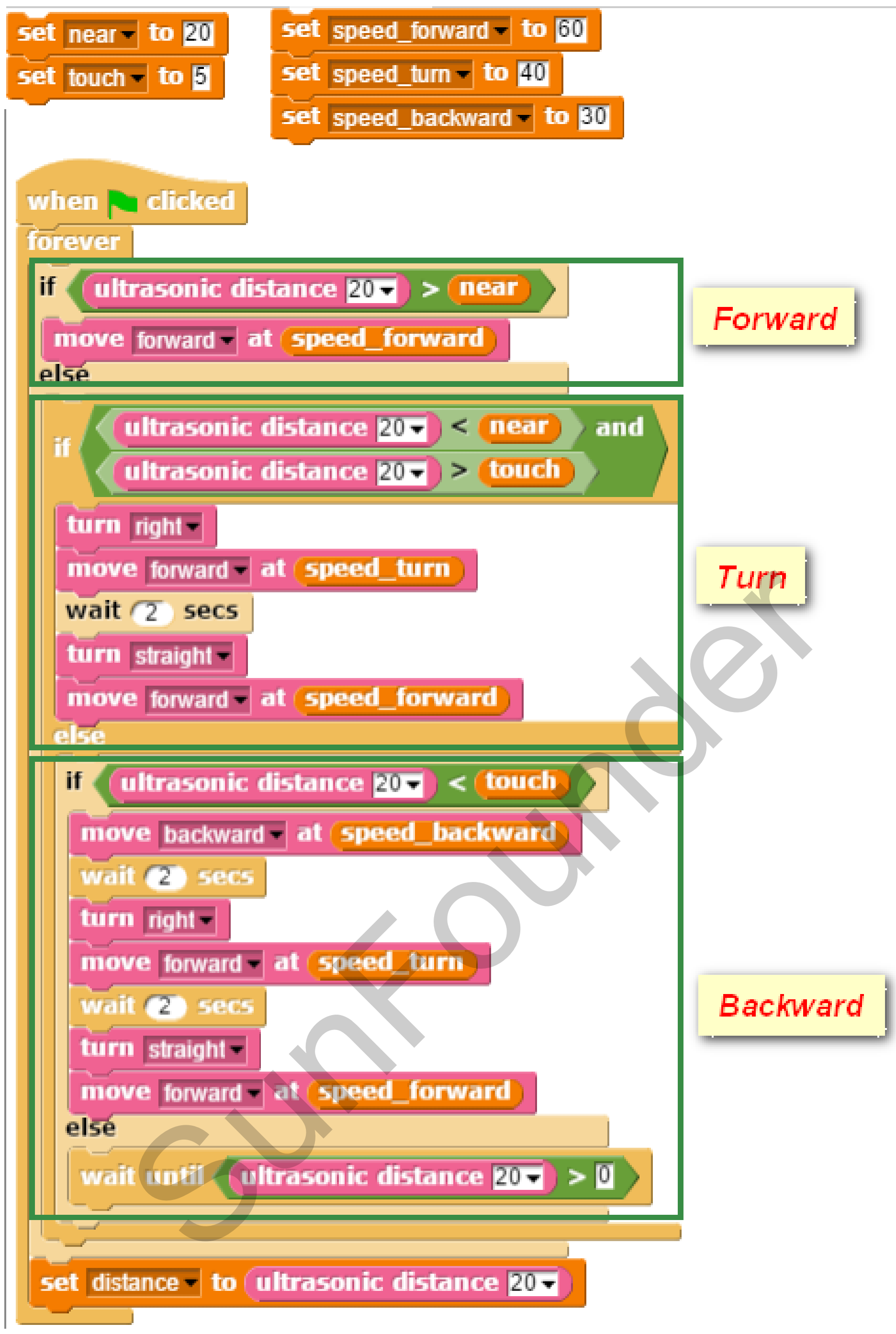
✧ Automatic Obstacle Avoidance

When the obstacle is close, the car turns around.

If the obstacle is too close that the car cannot even turn around, it will reverse first.

When the obstacle gets farther and does not hinder the car, it will go straight.




```
set near to 20
set touch to 5
set speed_forward to 60
set speed_turn to 40
set speed_backward to 30

when clicked
forever
    if ultrasonic distance 20 > near
        move forward at speed_forward
    else
        if ultrasonic distance 20 < near and ultrasonic distance 20 > touch
            turn right
            move forward at speed_turn
            wait 2 secs
            turn straight
            move forward at speed_forward
        else
            if ultrasonic distance 20 < touch
                move backward at speed_backward
                wait 2 secs
                turn right
                move forward at speed_turn
                wait 2 secs
                turn straight
                move forward at speed_forward
            else
                wait until ultrasonic distance 20 > 0
    set distance to ultrasonic distance 20
```

Click the green flag at the top right corner to run the whole program of automatic obstacle avoidance.

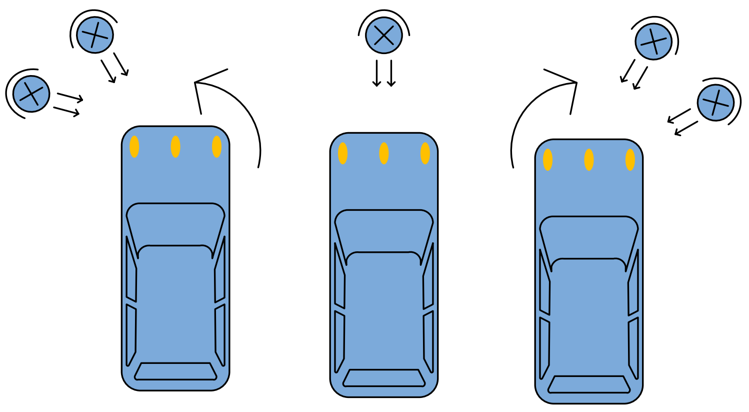
Place the car on the ground and power it on. It will go freely. When encountering an obstacle (you can set some gadgets around its path randomly on the floor), it will turn around itself. And if it's too close to the object, it will go backward to a safe distance, turn a direction and then move forward again when the obstacle is no longer in its near front.

✧ Automatic Light Following

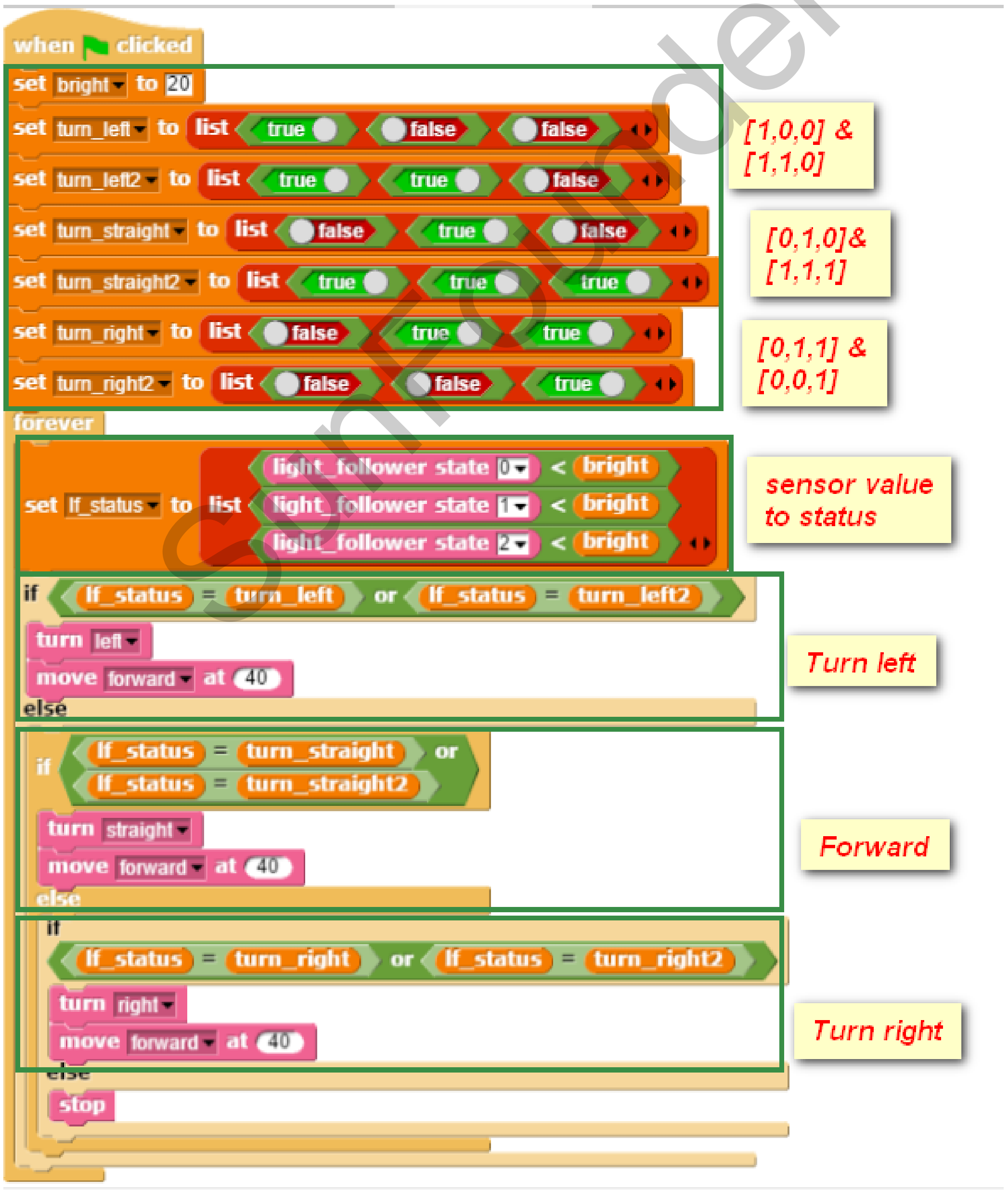
If there's light shone on the middle probe, the car will go straight.

If the left probe senses light shining, the car will turn left.

Similarly if the probe on the right side detects light, the car will turn right.



Turn left	Forward	Turn right
[1,0,0]or[1,1,0]	[0,1,0]or[1,1,1]	[0,1,1]or[0,0,1]



```
when [flag] clicked
set bright to 20
set turn_left to list true false false
set turn_left2 to list true true false
set turn_straight to list false true false
set turn_straight2 to list true true true
set turn_right to list false true true
set turn_right2 to list false false true
forever
    set lf_status to list (light_follower state 0 < bright)
                          (light_follower state 1 < bright)
                          (light_follower state 2 < bright)
    if lf_status = turn_left or lf_status = turn_left2
        turn left
        move forward at 40
    else
        if lf_status = turn_straight or
           lf_status = turn_straight2
            turn straight
            move forward at 40
        else
            if
            lf_status = turn_right or lf_status = turn_right2
                turn right
                move forward at 40
            else
                stop
```

[1,0,0] & [1,1,0]

[0,1,0]& [1,1,1]

[0,1,1] & [0,0,1]

sensor value to status

Turn left

Forward

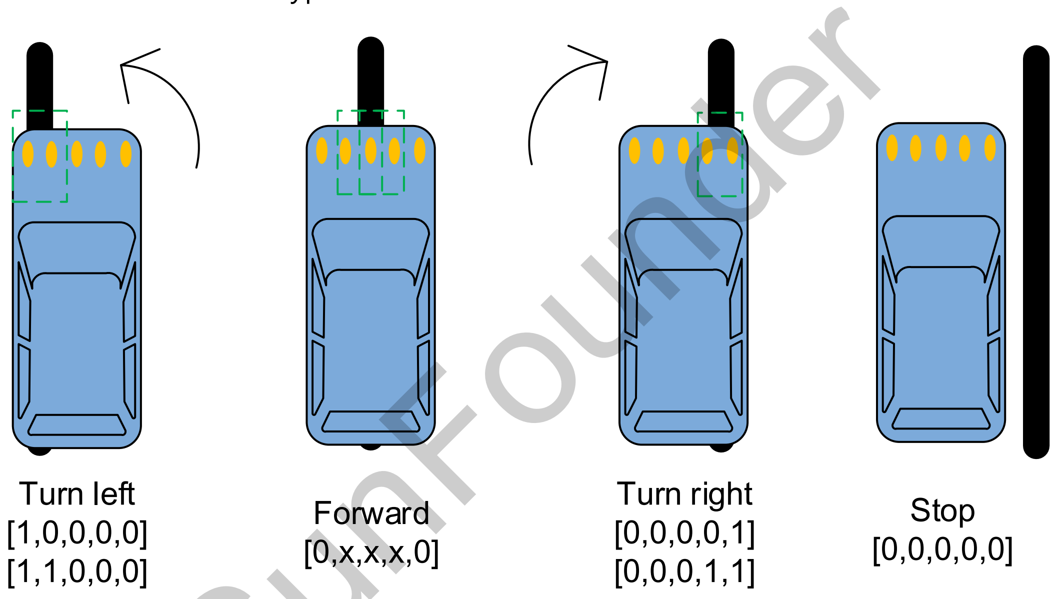
Turn right

Click the green flag at the top right corner to run the whole program of automatic light tracking/following.

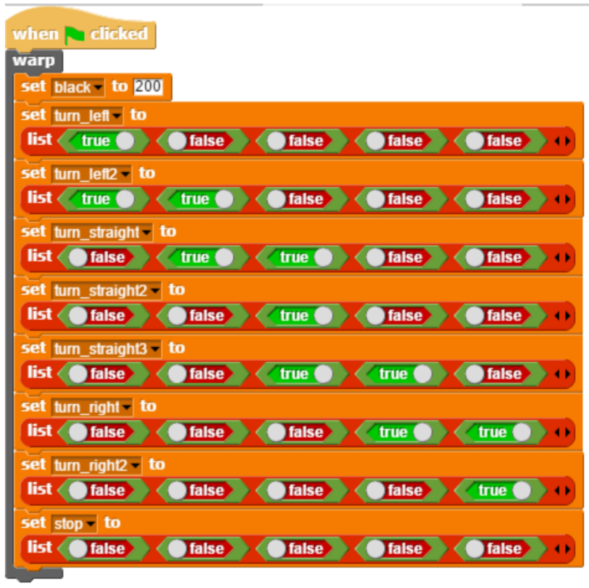
Place the car on the floor and use to a small torch, a flashlight to shine the car's sensors. Move the focus between the 3 probes of the Light Follower sensor. You can see when shining on the left probe, the car turns left; shine the light on the middle one, it goes straight; shine on the right-side one, the car turns right. Just like you're pushing the car with an "invisible hand", amazing!

✧ Automatic Line Following

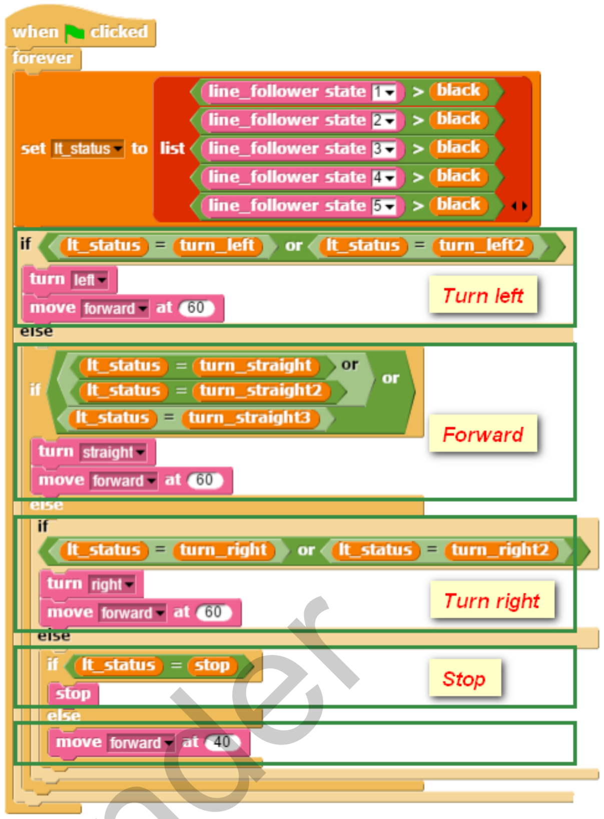
On the Line Follower sensor there are 5 probes, which are mapped to the 5 members on the list in code. They can have various performance in different combinations. Here we divide these circumstances into 4 types:



Based on the figure above, when the left probe of the sensor detects a black line , it turns left; when the probe on the right "sees" a black surface, the car turns right; when the middle probe reads the line, the car will go straight forward.



```
when [flag] clicked
warp
  set black to 200
  set turn_left to
  list true false false false false
  set turn_left2 to
  list true true false false false
  set turn_straight to
  list false true true false false
  set turn_straight2 to
  list false false true false false
  set turn_straight3 to
  list false false true true false
  set turn_right to
  list false false false true true
  set turn_right2 to
  list false false false false true
  set stop to
  list false false false false false
```



```
when [flag] clicked
forever
  set lt_status to list  line_follower state 1 > black
                         line_follower state 2 > black
                         line_follower state 3 > black
                         line_follower state 4 > black
                         line_follower state 5 > black
  if  lt_status = turn_left  or  lt_status = turn_left2
    turn left
    move forward at 60
  else
    if  lt_status = turn_straight  or  lt_status = turn_straight2  or
        lt_status = turn_straight3
      turn straight
      move forward at 60
    else
      if  lt_status = turn_right  or  lt_status = turn_right2
        turn right
        move forward at 60
      else
        if  lt_status = stop
          stop
        else
          move forward at 40
```

Now place the car on the map you drew with black marker pen or pasted via black tape. Click the green flag and the car will then run following the line. You can find the track map on our website *sunfounder.com* and download:

循线赛道地图的下载链接

Wow, you've come so long and really good job that you made it! So, that's all for the tutorial of **PiCar-S**. You can try more possibilities with **Dragit** for the robot and make your own program. Have fun!

Further Learning

Since the left and right angles of the front wheels are fixed, the car may not follow the lines quite smoothly. It may wind its way along the black line, or run out of track easily. These are inevitable since the Dragit is designed to make control simpler and programming laymen have an easier way to start learning.

If you want better performance of the car's movement after you've mastered most of the basic skills, you can download the package on the LEARN page on our website sunfounder.com and try to modify and run the code. That would enable more possible gameplays with the car, as well as provide smooth and efficient control.