Profile of Dragit

For how to assemble the PiSmart Car, refer to the video tutorials:
https://www.sunfounder.com/video/category/?cat=60

Instead of coding PiSmart directly in Python, you can also do it in a graphical/visual programming software - Dragit, developed based on Snap! (check http://snap.berkeley.edu/ for more). What's good about Dragit is that, the programming process is visible with graphics. So you can just drag and drop the blocks (unit in the software) to make the code to control the car, needing no knowledge of any programming languages or syntaxes.

So now, let's explore the wonderful programming world in PiSmart, with 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:

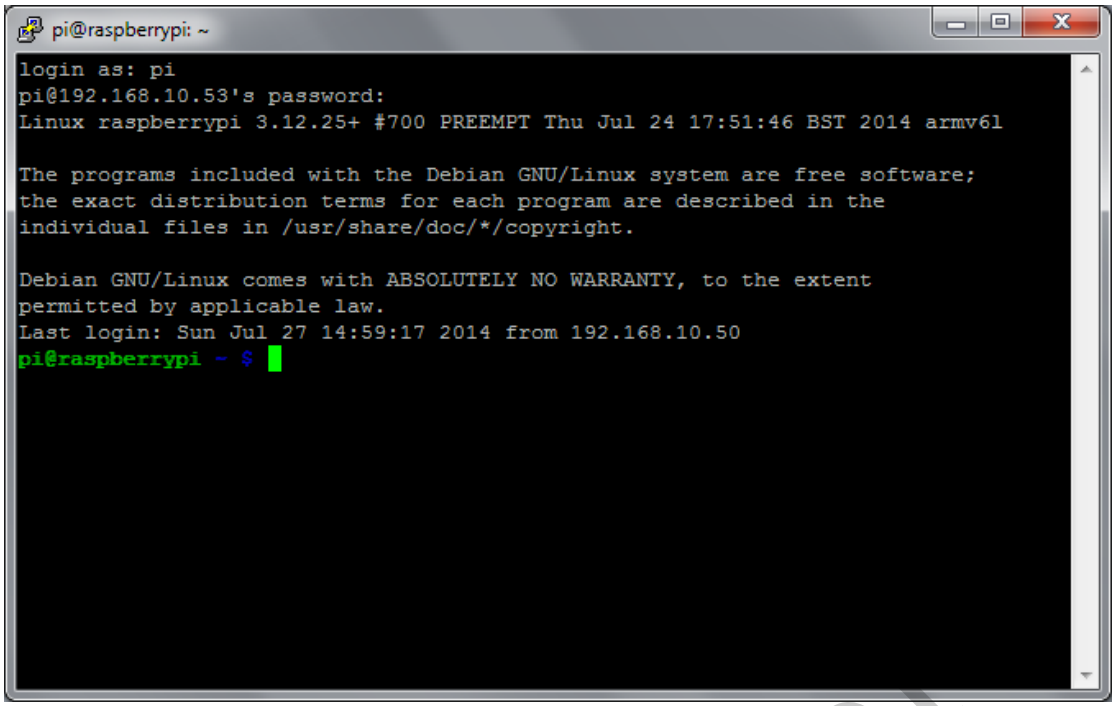
https://www.raspberrypi.org/documentation/remote-access/ip-address.md

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

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

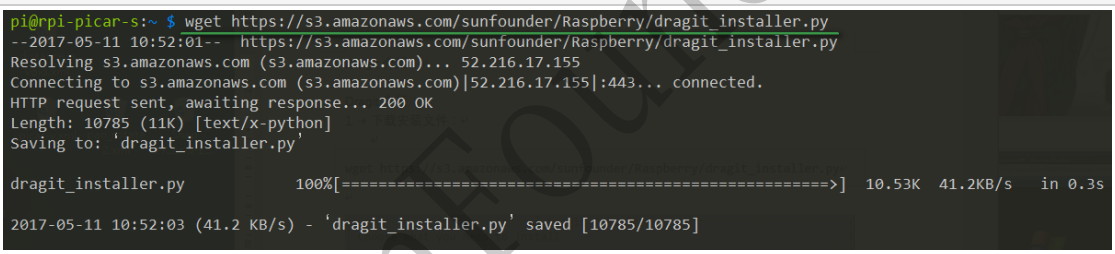
For VNC: https://www.raspberrypi.org/documentation/remote-access/vnc/README.md

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



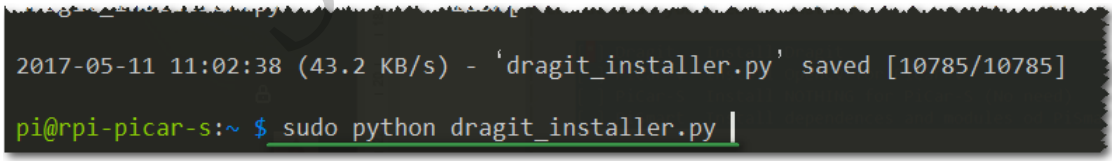
1. Type in the command below:

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



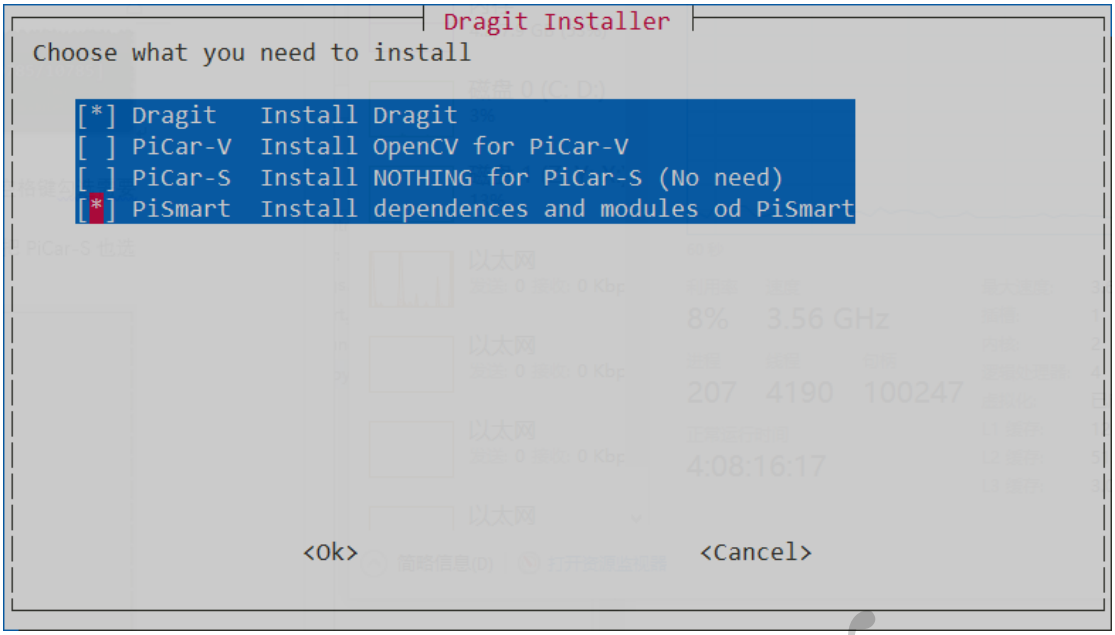
2. After download is done, type in the command to run:

```
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 **PiSmart**, so select it too. Then press Enter to confirm.



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.

```
Best match: peppercorn 0.5
Processing peppercorn-0.5-py2.7.egg
peppercorn 0.5 is already the active version in easy-install.pth

Using /usr/local/lib/python2.7/dist-packages/peppercorn-0.5-py2.7.egg
Finished processing dependencies for SunFounder-PiSmart==1.0.0
all done, enjoy it
Error
pi@raspberrypi:~ $ sudo reboot
```

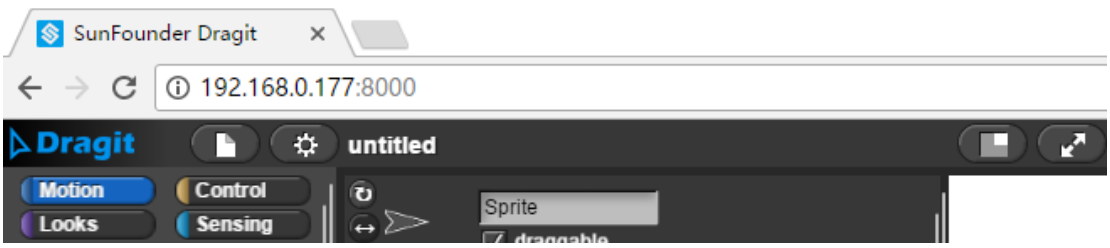
When the prompt appears as shown above, it means the installation is done.

After installation, reboot:

```
sudo reboot
```

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

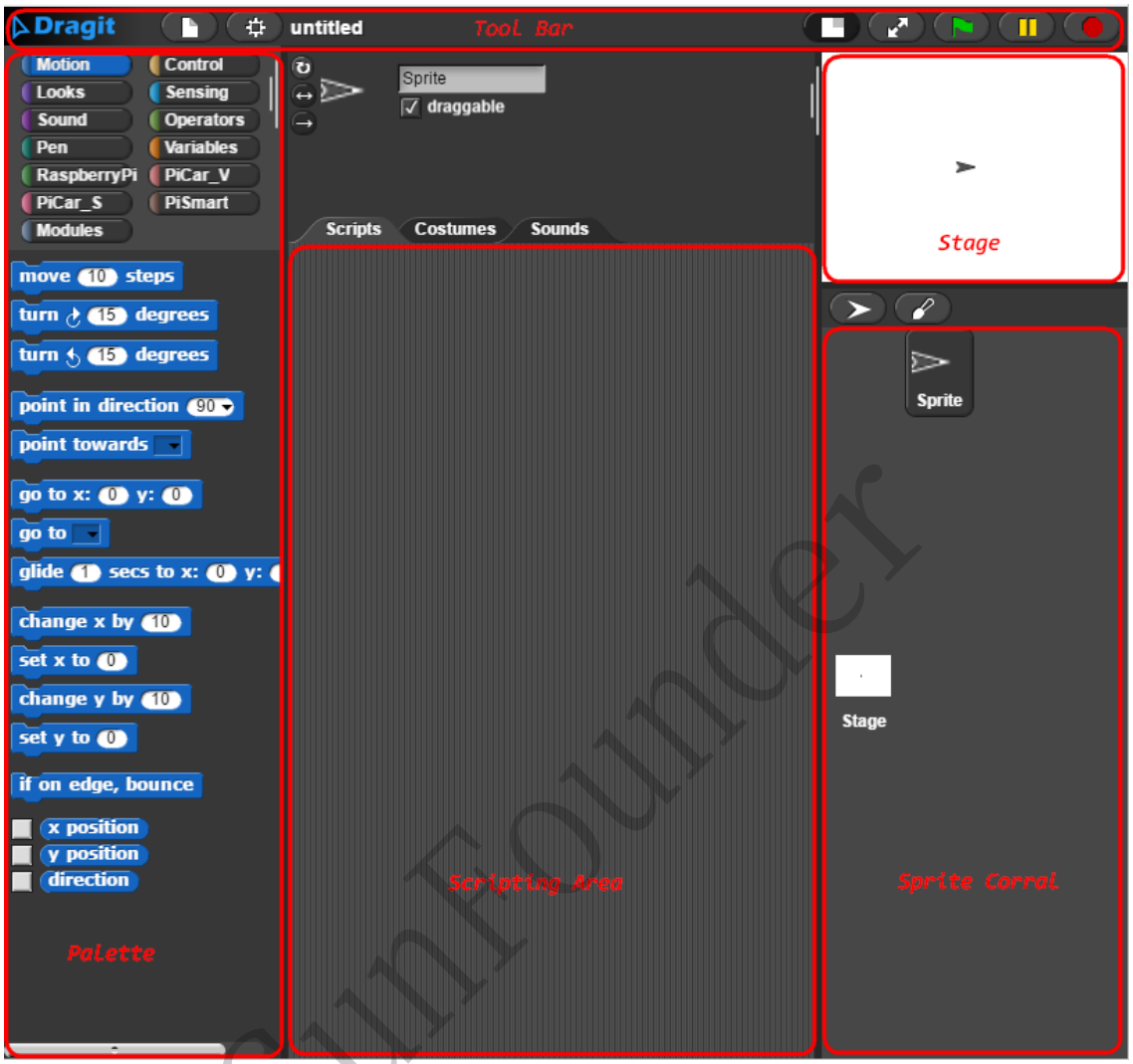
Example: 192.168.0.177:8000



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 PiSmart, we'll focus on the category of PiSmart, basics of using Dragit, and how to control PiSmart in this software.

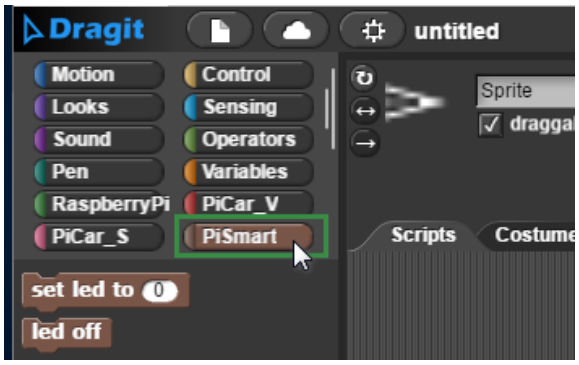
Mission 1: Drive the Car

To use PiSmart on the Dragit, first to learn is how to control the car to go forward. It's one basic movement for this robot. Similar such movements like turning are done in the same way.

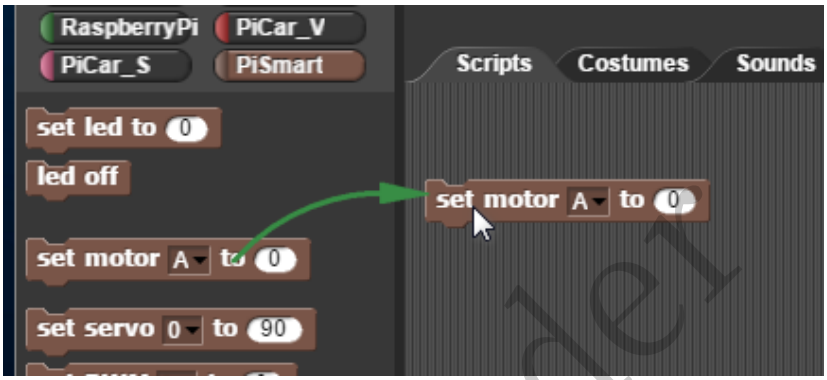
Before operation, in case the car runs out of the desk and falls off, we prop it off the surface.

And then take these steps:

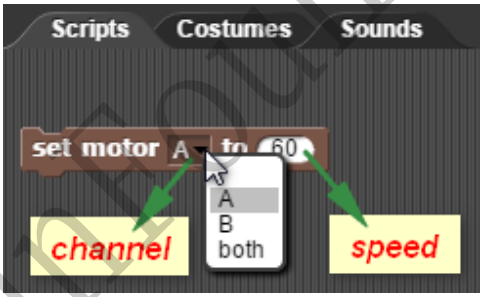
1 · In the **palette** area, click to select the PiSmart category.



2 · Then click and drag set motor A to 0 into the **scripting** area.



3 · Enter values in the slot:



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

For the left one, *channel* is to set the motor you want to control. Click the input of direction by the **triangle** sign and select a value at the drop-down list. Click A, meaning you'll control the motor at MotorA. Then click the right-side parameter, *speed*: enter a number ranging between -100 and 100. The larger value you enter, the faster the motor spins. Positive or negative value indicates the motor turns clockwise or counter-CW. So now your block defines the motor, its speed and the rotational direction.

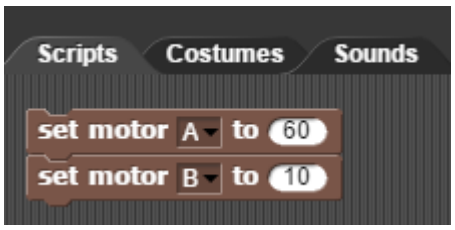
4 · Click this block and see what will happen to the car. Yes! The motor at MotorA starts to spin.

The two motors on the PiSmart car at two sides connect to port MotorA and MotorB. We can select the channel both in the block, so two channels output values at the same time, thus controlling both motor A and B at one time.

For example, click both on the list and enter 60 in the *speed*, meaning outputting a speed of

60 to both motor A and motor B. Click the block then and the car will move forward!

5 · If you want to let the car turn a direction, you can combine two blocks to do so:



Set different values to the speed. When the wheel at one side spins faster than the other, the car will take a turning.

6 . To stop the car, just set the *speed* of **both** motors as 0:

7 . Set the value of *speed* as negative for **both** motors, so the car runs backward:

Mission 2: Let It Speak

Since the car can walk now, let's check how to let it speak!

A speaker is integrated on **PiSmart**. We can enable it to speak for speech message feedback.

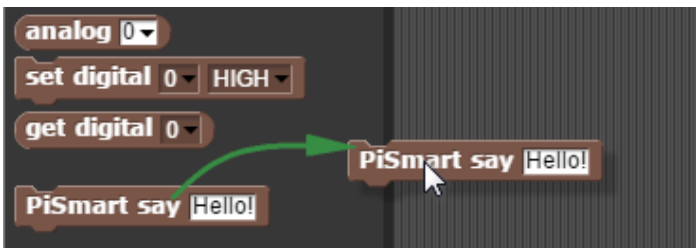
This function applies the block 

to produce speech.

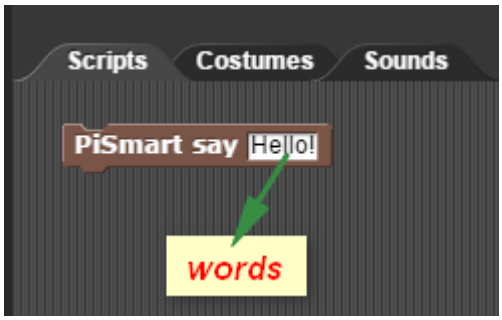
1 . Click the **PiSmart** category in the palette area. Skip it if you're already there.

2 . Click 

and drag it into the scripting area:



3 · So now in the scripting area there's already a block to control the PiSmart to speak. Let's check how it is done with the block!



In the block, you can click the slot and enter what you want the PiSmart to say. By default it's "**Hello!** ".

Click the block and you'll hear what's said by the speaker of the PiSmart, and that's exactly what's been entered in the slot. It's, **Hello!**

Now try your own words! Multiple words are supported. Just enter and click the block. See, what you enter, what the PiSmart will say. It's perfectly your mouth!

OK, simply done. Let's move on to next adventure!

Mission 3: Make the Car Understand You

Now, the car can speak. Next, we need to make it understand our commands.

There is a MIC in the PiSmart as its "ear". You only need to drag the corresponding blocks to make it get your commands (but need to be preset).

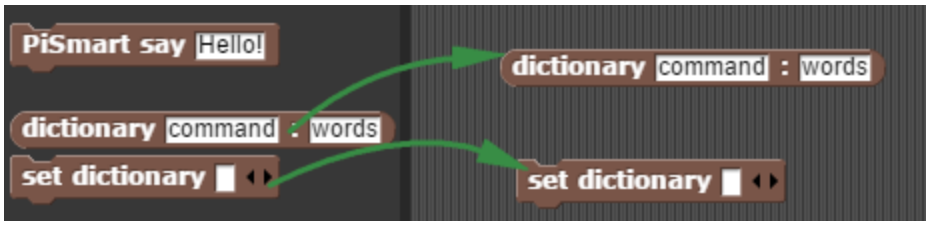
Set Dictionary:

Before using the hearing function, we need to set a dictionary first. What it does is to "tell" the PiSmart to recognize the words *xxx* as the *xxx* command.

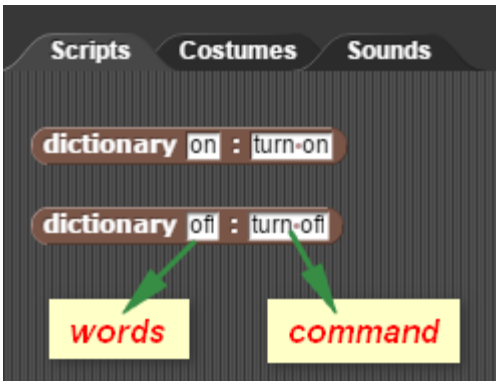
For example, when you say ***turn on*** (as called words), the PiSmart can understand it means **on** (as deemed command); when you say ***turn off*** (words), it knows it's ***off*** (commands).

Use these two blocks to set a dictionary: set dictionary ▢ ◂▸ & dictionary command : words.

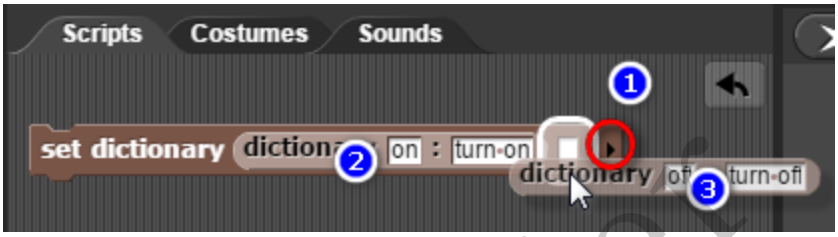
1 . Drag the two blocks below from PiSmart into the scripting area:



2 . In the block dictionary ▢ : ▢, enter the *words* and *command*. Since I want to use two commands, duplicate one more block. Enter values for turning off.



3 . Then put these two blocks into the slot of  - click the triangle icon to add another slot:



4 . Click this combined block. Then in the back-end system a file will be generated recording the mapping between the words and the command. Call the .sps file.

When the file is created, the dictionary is done. Now we're ready for the hearing function.

Use Hear Blocks:

For the hearing function, blocks  ,  , and  are available.

1 . Click the category and drag these blocks from PiSmart into the scripting area:

2 . Say "turn on" to the PiSmart and then click each and you can see a bubble is popped up at the right side:

Now let's probe into details of each block:

heard - When the PiSmart "hears" words recorded in the .sps file, the bubble will return **true**, otherwise **false**. So only recorded words will be heard and corresponding result returned. For instance, in the example above, since only "*turn on*" and "*turn off*" are recorded in the file, only when you speak these two phrases, will the command **true** be returned.

hear result - It returns what the PiSmart heard. Only when the words heard are included in the .sps file, the corresponding result will be returned. For example, previously only the commands on and off have been recorded, so only when you speak "*turn on*" to the PiSmart, *on* will be returned; say "*turn off*", *off* will be returned.

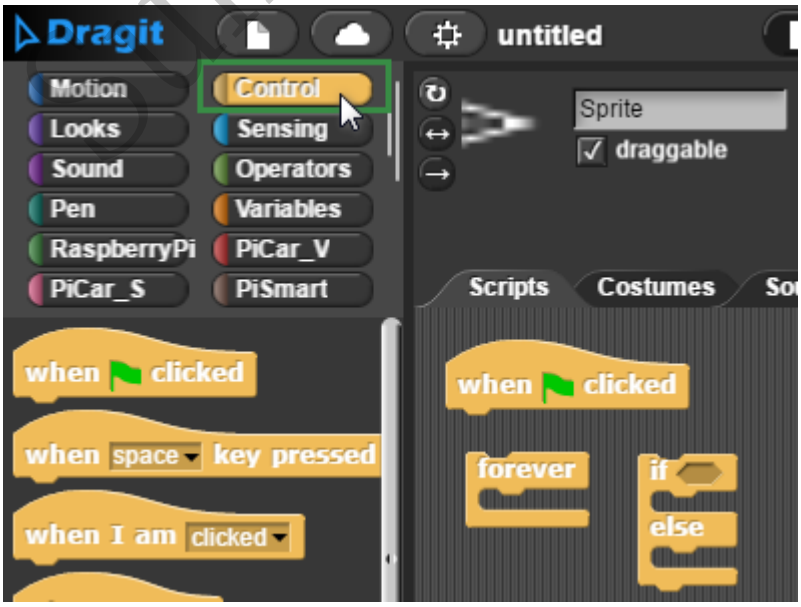
[ ] is heard - It verifies whether the command just entered has been heard. If yes, it will return **true**, otherwise **false**. For example, if hear result returns a command *on*, on is heard will return **true**, when ofi is heard will give back **false**.

Since the Hear block returns the results of **true** or **false**, generally the **Logic control blocks** will be used together for better performance of speech control.

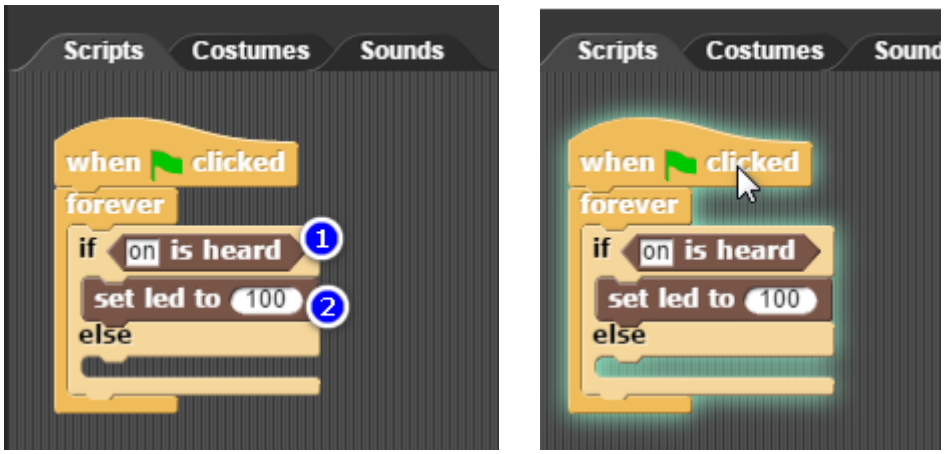
Example: Use Speech to Control LED

Previously we've set a dictionary, with which we can make a program to control lighting up and out the LED.

1. Click the Control category and drag the following blocks into the scripting area:



2. Drag the block set led to 0 and drop to combine these blocks, drag on is heard to the slot after if:



3. Click this combined block and say "turn on" to the PiSmart. Wow, the LEDs on the car just light up!

   To understand: When the PiSmart receives the command for the words "turn on", block ① above will return **true** and the program will run the brightness setting in block ②; If no command is received, since there's no content under else, nothing will happen. Therefore, after the LEDs light up, you need to click the led off block to turn them off.

4. Click the combined block again and the program will stop running.

5. Drag another if block and drop it under else, so after the PiSmart "hears" "off", LEDs will be turned off.

   Click the chunk block again. This time try to say "turn on" and "turn off" to the PiSmart and observe how the LEDs act. Interesting control, isn't it?

So when you say "turn on" to the robot, the LEDs will light up accordingly. If you say "turn off", LEDs will dim then. But if you say other things, since no more is written in the combined block, it won't do anything and just stay the status quo.

Mission 4: Control the Car via Speech

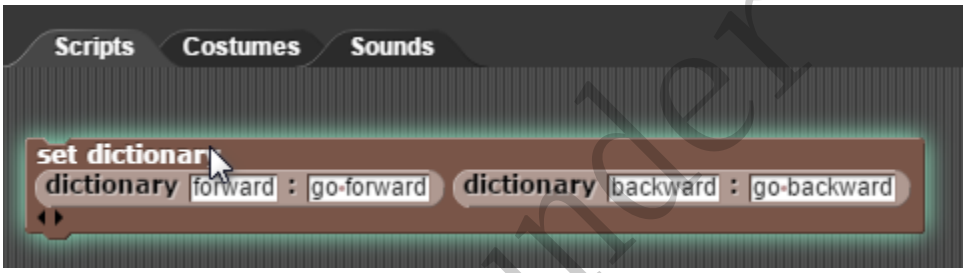
After learning in the previous mission, let's combine the knowledges to make a speech control

program by reviewing.

So here's our aim: When you speak "go forward" to the PiSmart car, it goes forward after replying you "I go forward"; If you say "go backward", it does so and replies "I go backward". Can't wait to make it!

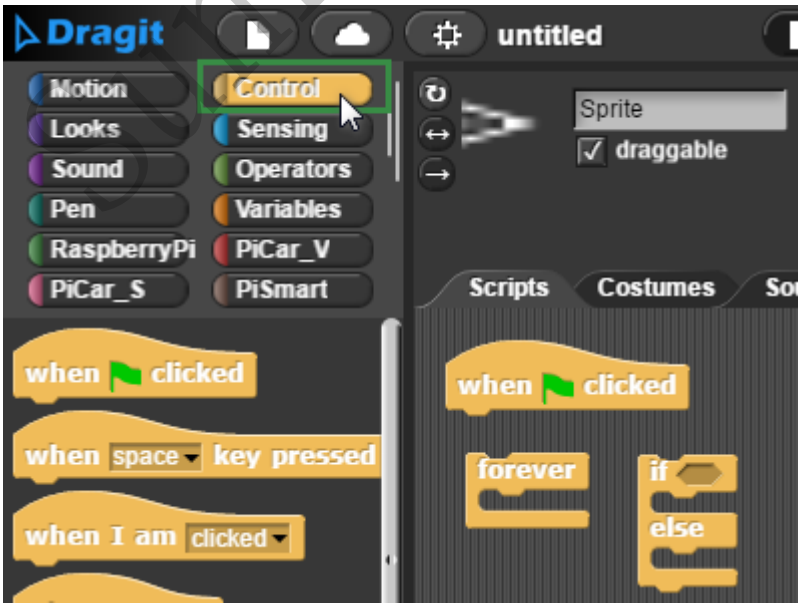
1．Set a dictionary:

Enter command and words as we did in last mission:

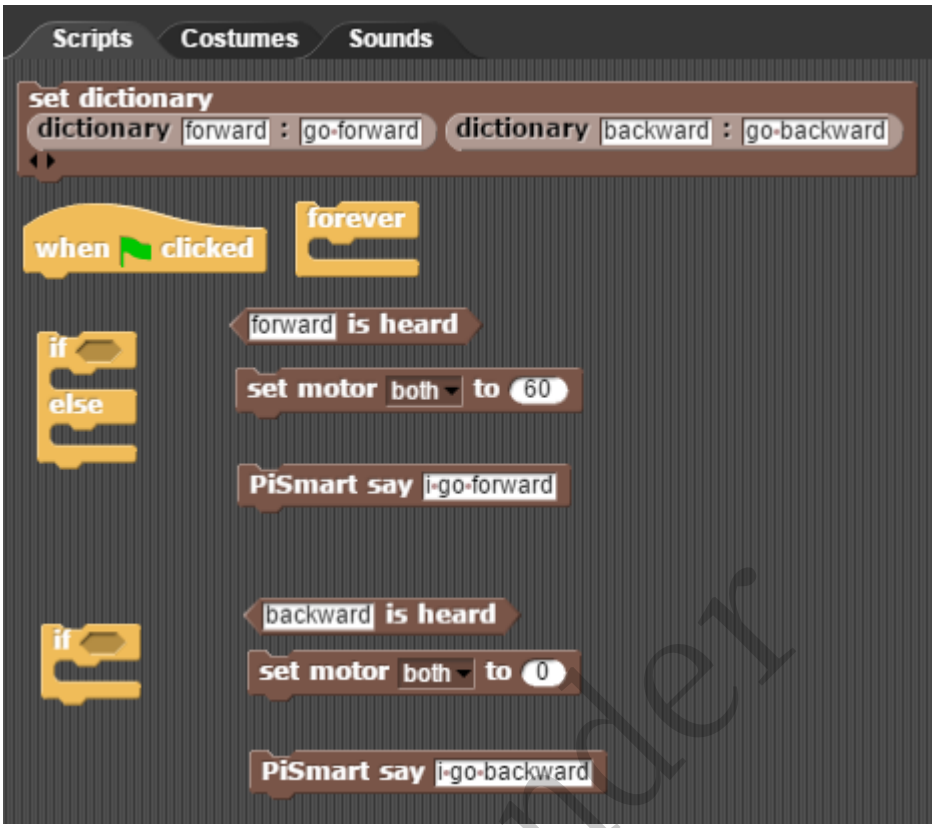


Drag and drop the dictionary block into the slot in set dictionary one. Click the chunk and setting will begin.

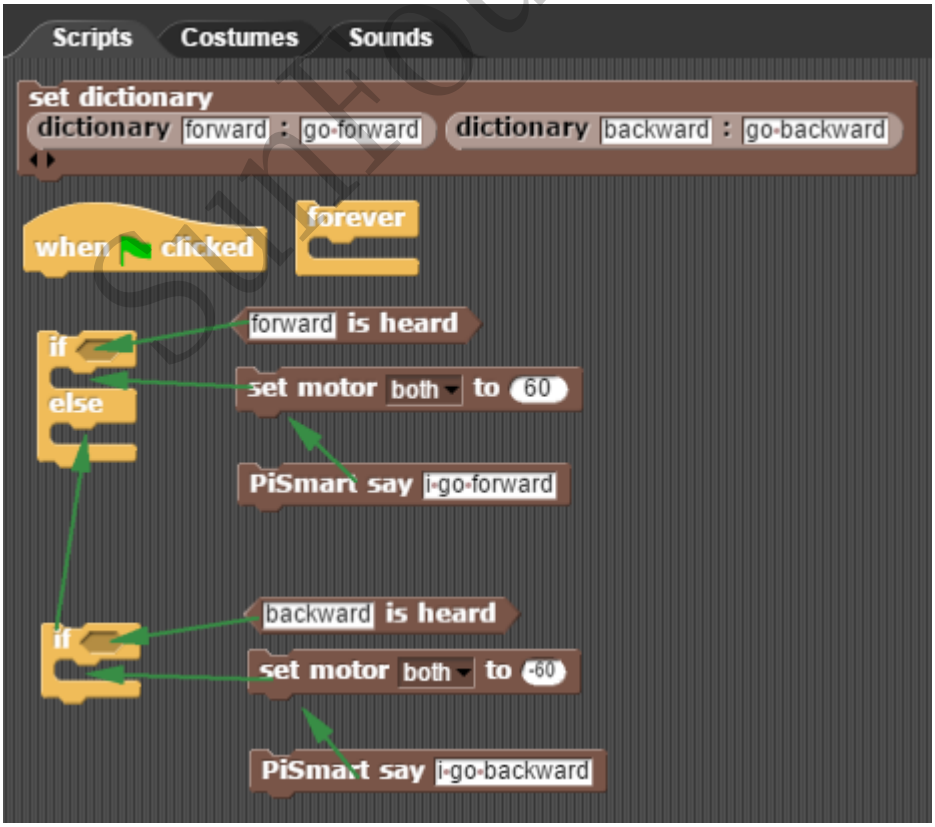
2．Drag the following blocks from the Control category:

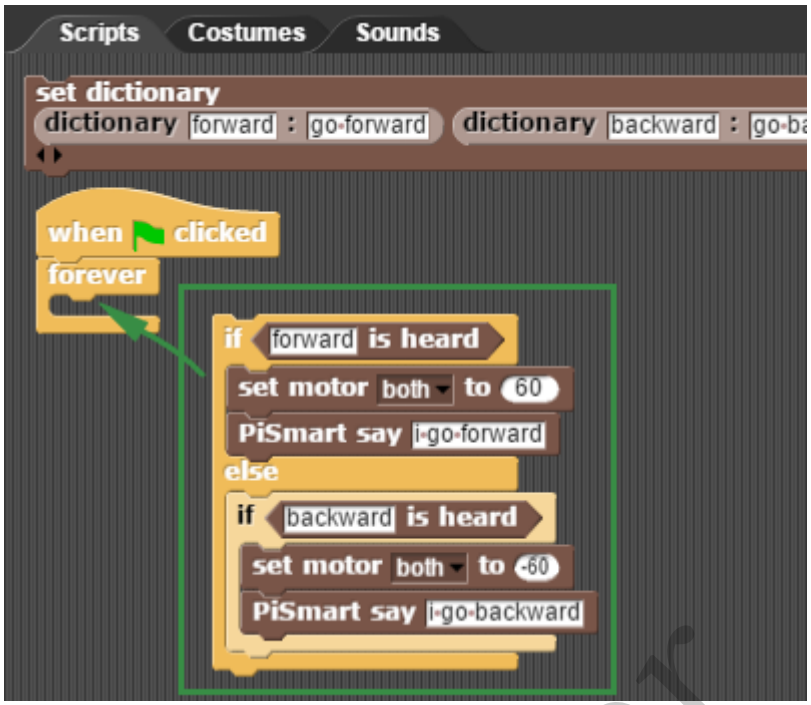


3．Drag the blocks below in the PiSmart category:



4．Combine the blocks, and change the value:





5 . Click the green flag at the top right corner to run the program:

6 . Now the program is running. You can say something to the MIC on the PiSmart. Try "go forward". If it recognizes the command, it will do as you ordered, run forward. Sometimes if it's too noisy, or you pronounce the words not so clearly, PiSmart may not understand you. Try again then. When it hears you, it'll answer "I go forward" before going.

In this example, we've included what's learnt before and made a comparatively complex task of speech control. But I bet you feel it easier than imagined as we did it in the Dragit. And it's great fun! This is no doubt a wonderful way to program the PiSmart Car in Dragit.

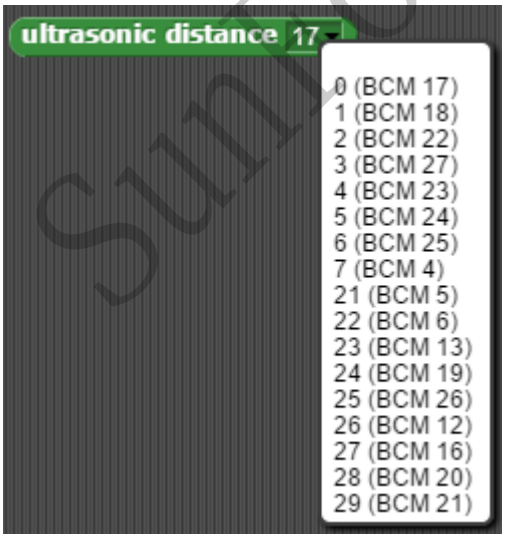
Mission 5: Get Ultrasonic Distance

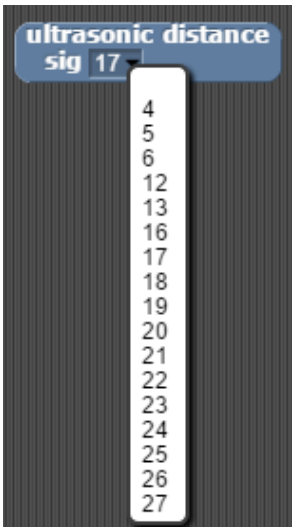
To make the robot car detect the ambient environment by the sensors, and for its autopilot, we can use the ultrasonic obstacle avoidance and line following sensors included in the package.

Find the blocks in the Modules category:

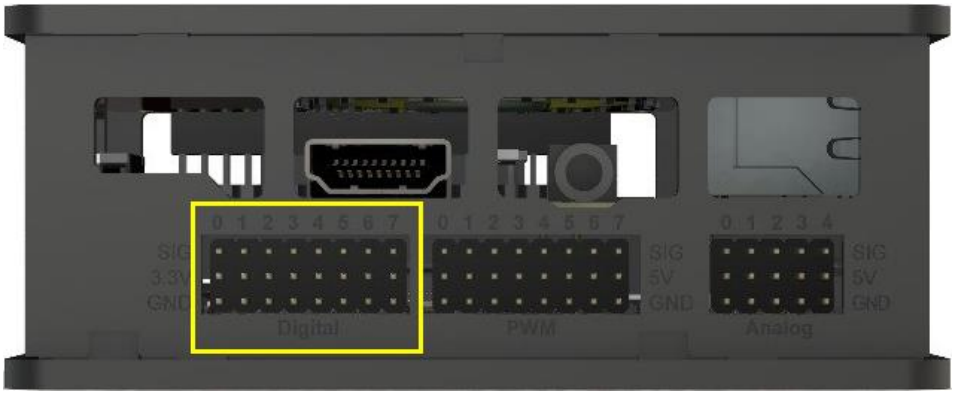
Before using the block, please make sure the module is connected correctly to the car. Otherwise the block will return -1.

The ultrasonic sensor can detect obstacles in front via the principle of ultrasonic waves. Then in programming, we can make the car take a turning to detour the obstacle or continue going straight based on the distance detected.





On the drop-down list of the block, there are many channels for ultrasonic module connection. On the PiSmart they are digital channels 0-7.



Here we connect the module to channel 0 - connect the yellow wire to SIG0, red to 3.3V, and the black wire to GND. Then click 0 (BCM17) on the list of the block.

After wiring, click the block. And the distance value detected will be returned then.

Store Data Detected by Sensor

As described above, click the block and you can check the data detected by the ultrasonic sensor. Before using the data, we still need to store it.

Use a variable to store the distance data: Variables

1 · Click the Variables category.

2 · Click Make a variable to create a variable.

3 · A window will pop up. Enter a name for this variable.

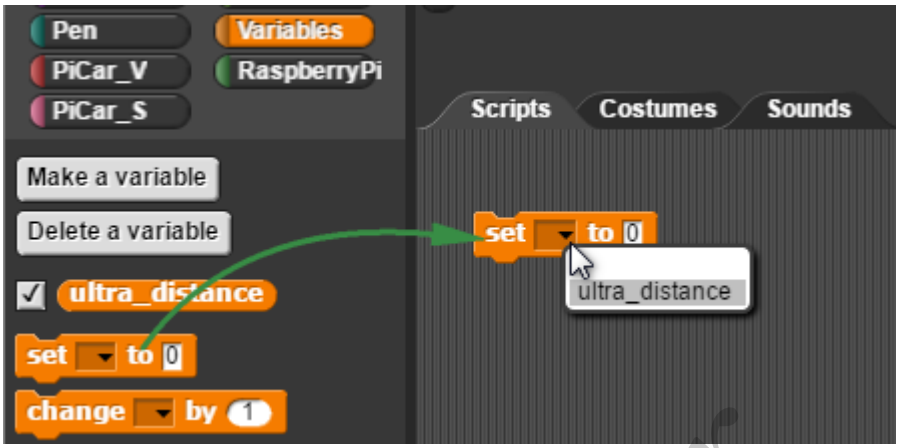
4 · Click **OK** after naming it. Then at the palette you can see the variable created.

Now you can use the variable, the ultra_distance block. Add another block 

to store the data detected by the sensor in the variable just created.

5. Click and drag 

into the scripting area, click the triangle icon to select ultra_distance:



6. Click the Modules category, drag 

to the slot in 

.

Click the combined block to run once, so the data detected by the sensor is written to the ultra_distance variable and stored.

On the **stage** area at the top right corner, you can see ultra_distance and the distance value it stores:

Place something like a book not far in front of the ultrasonic sensor, with the probes oriented to the book. Click the block. Observe the value of ultra_distance in **stage**. That's the distance detected by the sensor in the unit of cm.

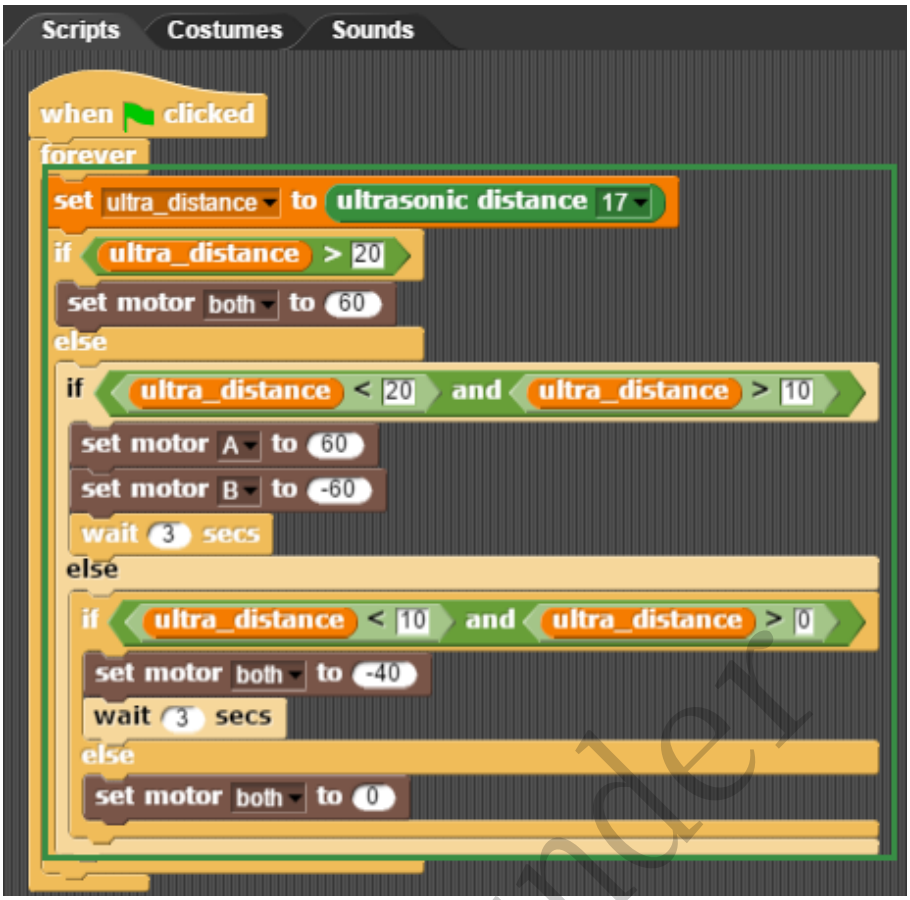
Move the book forward and backward and click the block again. Any change to the value?

Well, perfectly done if you get a smaller or larger value. But if **-1** is shown on **stage**, it means the detection fails. The possible reason may be wrong wiring of the sensor's pin **sig** to **0**, or the wrong **port** selected in the block. Go back to screen your blocks building! You can't be more careful in configuring these values.

OK, since everything goes well, now we can apply the variable ultra_distance in the logic judgement.

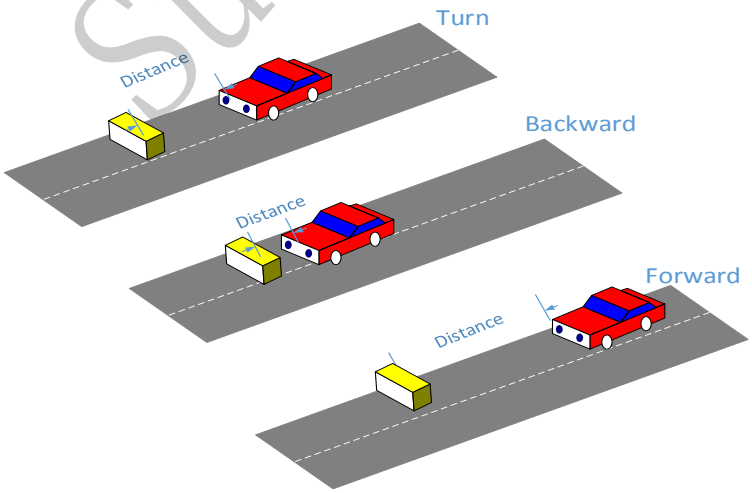
Here 4 if statements are used. The first one is to control the car to go forward when the distance is larger than 20cm; the second, turn a direction if it's between 10 and 20; the third one, make the car reverse if it's 0-10cm; as for the fourth, no specific condition is made, but it's for the case when an **error** occurs to the value and **-1** is returned: the robot will stop.

Now put this combined chunk under the forever loop and a start block:



Click the green flag on top and the whole block will start running. That's really a big chunk.

Move the obstacle - the book towards or take it away from the car, observe the value changes. You can see when the distance between the car and the book is larger than 20cm, the car goes forward; when the distance is 10-20cm, it turns a direction; when the distance is 0-10cm, the car backs. If you unplug the wire of the sensor, a value -1 will be returned and the car will stop.



Certainly you can add the speech communication and control into the program. So the car can report its error if any (e.g. loose sensor), thus you can fix it immediately. Try it yourself!

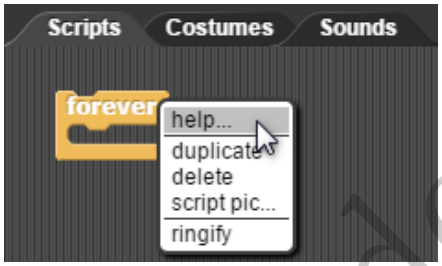
Tips: If you feel no clue of how to make it, find the example sketch in example.

Mission 6: Line Following

Connected with a Line Follower module, the PiSmart car can detect black lines and run tracking the path applying the principle of infrared reflection. On our website sunfounder.com you can find an e-map in PDF file under LEARN -> Get tutorials -> PiSmart, download and print it, and paste the black tape along the lines.

This Line Follower module connects with Raspberry via I2C: white wire to SCL, yellow to SDA, red one to 5V, and black wire to GND.

Previously we've gone through various blocks and how to apply them and no more details will be given here. So now let's check the example sketch. *Tips: If you're not clear about some block, right-click on the specific one and select* ***help*** *on the pop-up list.*

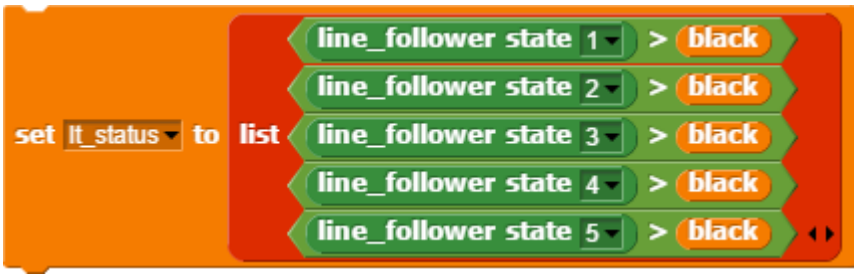


Find the complete example program in example.

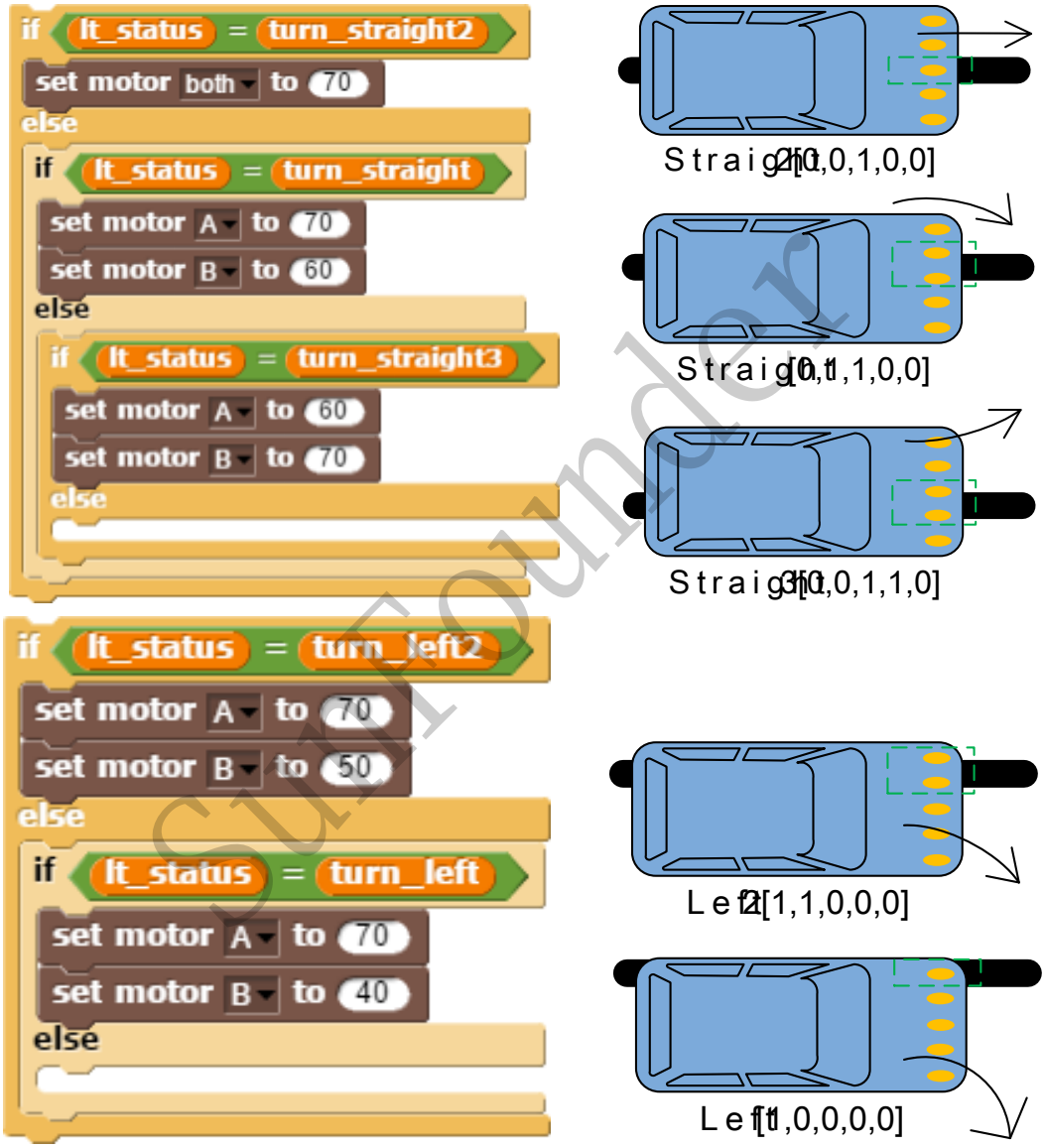
There are 5 probes on the sensor module, mapped with the 5 members in the list. Multiple combinations are possible. Store each channel into Variables, tell the status of the car by data detected by the sensor, and adjust the motors accordingly.

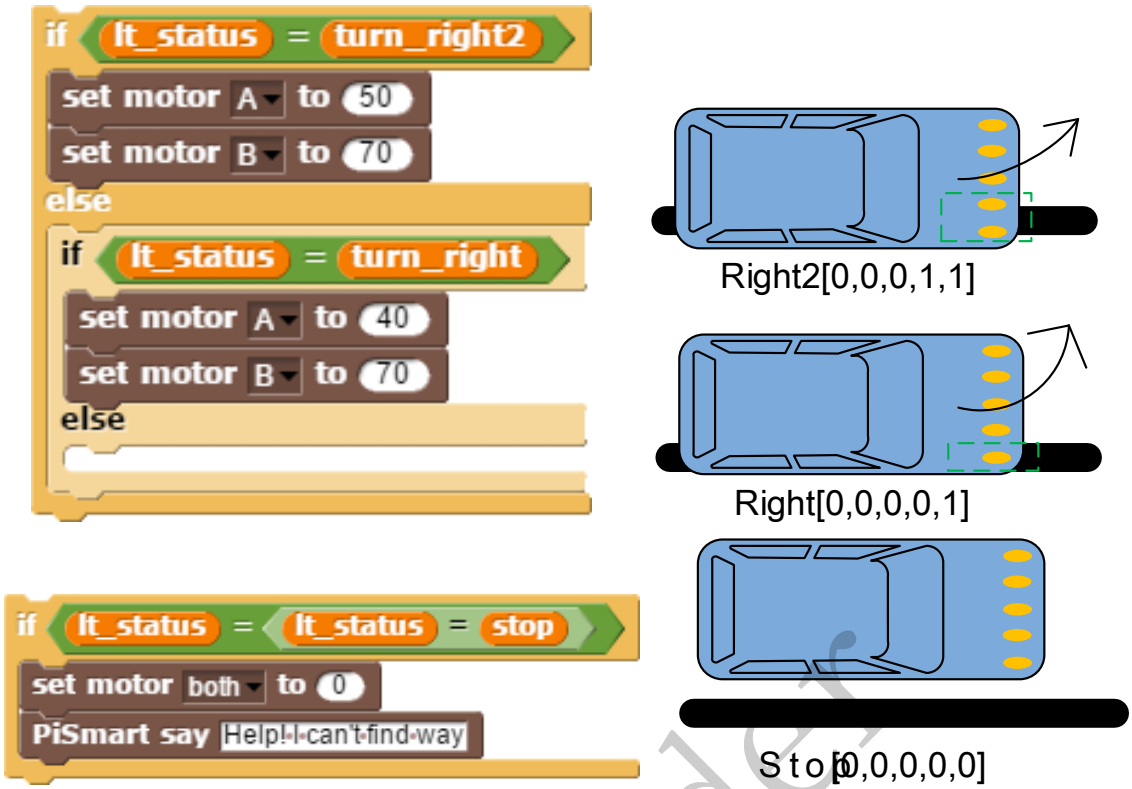
1 . Set the variables first to store the data which will be used in the judgements.

2 . Store the values acquired by the sensor, to be referred to later for the judgement.



3 . Decide actions to take based on the results of the judgement.





Then combine these chunks into one big block. Huge chunk then! But not so complicated as you may imagine, I guess, if you follow each step.

Future Learning

There are still many expansion ports on the PiSmart like spi, analog, UART, USB etc., enabling more DIY possibilities for the PiSmart Car - you can connect more sensors for other fun functions or control different actuators via PiSmart.

In the future tutorials, we'll add more fun projects with sensors, and more blocks will be complemented in the Dragit also in later version. Follow our website for more updates!

If you have any questions or ideas you want to share, visit our FORUM and post anything.