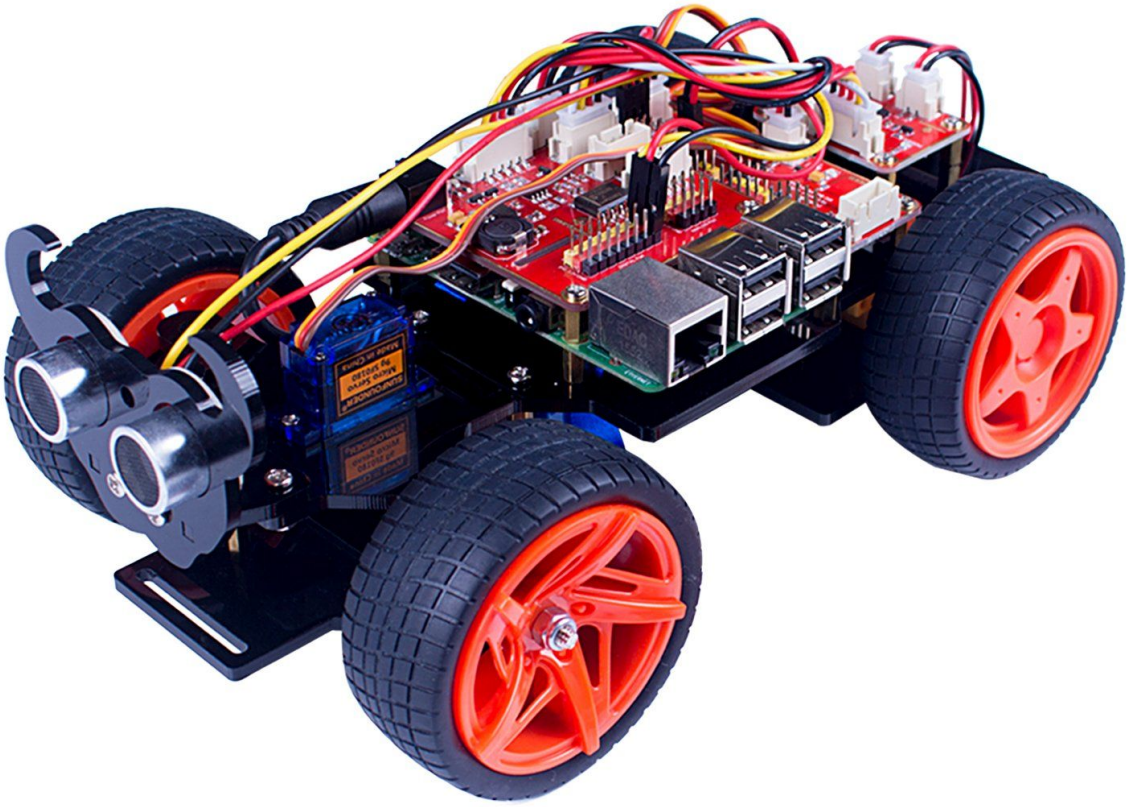
PiCar Workshop



By Joe Brown 8th May 2019

Using Command Line & Python

PiCar Setup

1. Switch on the picar - switch is on the top of the car near the front right wheel. You should see green lights

2. Download & save putty from bit.ly/picarputty (while still connected to normal wifi network - eduroam)

3. Now connect Laptop to the following wifi network:
   Wifi name: *hedgehogbot*
   Password: *hedgehog*

4. Open putty and type in the following
   Host Name: **this will be given to you by a helper** (192.168.1.X)
   Connection Type: SSH.
   Click Open. A black command window should pop up. If it doesnt, ask for help.
   (Helpers: IP is obtained from DHCP Client List on Hedgehogbot router)

5. In the command window enter the following:
   login as: *pi*
   password: *raspberry*

6. Now let's test the car is working. First the steering:
   Type *picar front-wheel-test* and press enter
   You should see the front wheel turning left and right. Ask for help if this isn't happening. Press **ctrl+c** to stop this.
   (Helpers: config file: **SunFounder_PiCar/picar/config** turning_offset -30 is most left, 30 most right - copy the config file to root.)

7. Now let's test the rear wheel drive. **Make sure the car is on the floor, not on the table!**
   Type *picar rear-wheel-test* and press enter
   You should see the car go forwards and then backwards then stop.
   (Helpers: config file: **SunFounder_PiCar/picar/config** forward_A & B = 0/1 to switch back motor rotation direction- copy the config file to root)

Writing Code for PiCar
We're going to write a python script to get the picar to move with commands.

1. Create a new python script called 'drive' by typing *nano drive.py*
   Nano is a text editor. You're now editing a new python script called drive.py

2. You need to add a few setup lines to the code, type the following into the file:

   ```
   import time
   import picar
   fw = picar.front_wheels.Front_Wheels(db='config')
   ```

   Make sure you've got that last line right with the capitals and underscores.

3. You're now all set up for controlling the front wheels! Add the following to your script below the setup lines:

   ```
   fw.turn_left()
   time.sleep(1)
   fw.turn_straight()
   ```

   Can you work out what it will do? time.sleep is a delay. fw controls the front wheels.

4. Now let's test your script. To exit press **ctrl+x**. It will ask if you want to save, press **y**.
   It will ask if you want to keep the file name, press **Enter**.
   Now to test the script, type: *python drive.py*
   Hopefully if your script works the picar's front wheels will turn left then back to center.

5. Let's add the back wheel setup. Edit the python script again by typing *nano drive.py*
   Add the bw setup to the file at the top, (previous setup lines in grey):

   ```
   import time
   import picar
   fw = picar.front_wheels.Front_Wheels(db='config')
   bw = picar.back_wheels.Back_Wheels(db='config')
   ```

6. Now let's code some back wheel controls around our front wheel controls
   (previous code in grey):

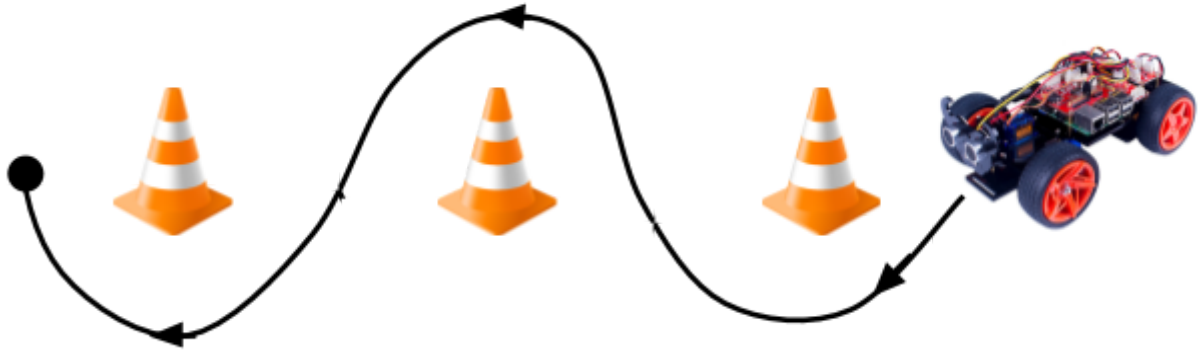
   ```
   bw.speed = 40
   bw.forward()
   fw.turn_left()
   time.sleep(1)
   fw.turn_straight()
   bw.backward()
   time.sleep(1)
   bw.stop()
   ```

Save the script (**ctrl+x**, **y**, **enter**) and run it (*python drive.py*) you should see the car go forward-left, then backward. Ask for help if it's not running how you think it should.
(Helpers: copy config file to location of participant's script if backwards/forwards are wrong way round)

7. Here are all the drive commands you can use top drive the car:

command	**What does it do?**
`fw.turn_left()`	Turn front wheels left
`fw.turn_straight()`	Turn front wheels to the middle
`fw.turn_right()`	Turn front wheels to the right
`fw.turn(X)`	Turn front wheels by X where X is an angle: 90 is straight, 45 is max left, 135 is max right
`bw.speed = X`	Back wheels will spin by X speed where X is a number: 1 is minimum speed, 100 is max speed.
`bw.forward()`	Spin back wheels forward by speed above
`bw.backward()`	Spin back wheels backward by speed above
`bw.stop()`	Stop back wheels
`time.sleep(X)`	Delay next command by X seconds

We've set up an some mini traffic cones for you to try navigate around using your python script. Edit it and see if you can code the picar to navigate the track like so:



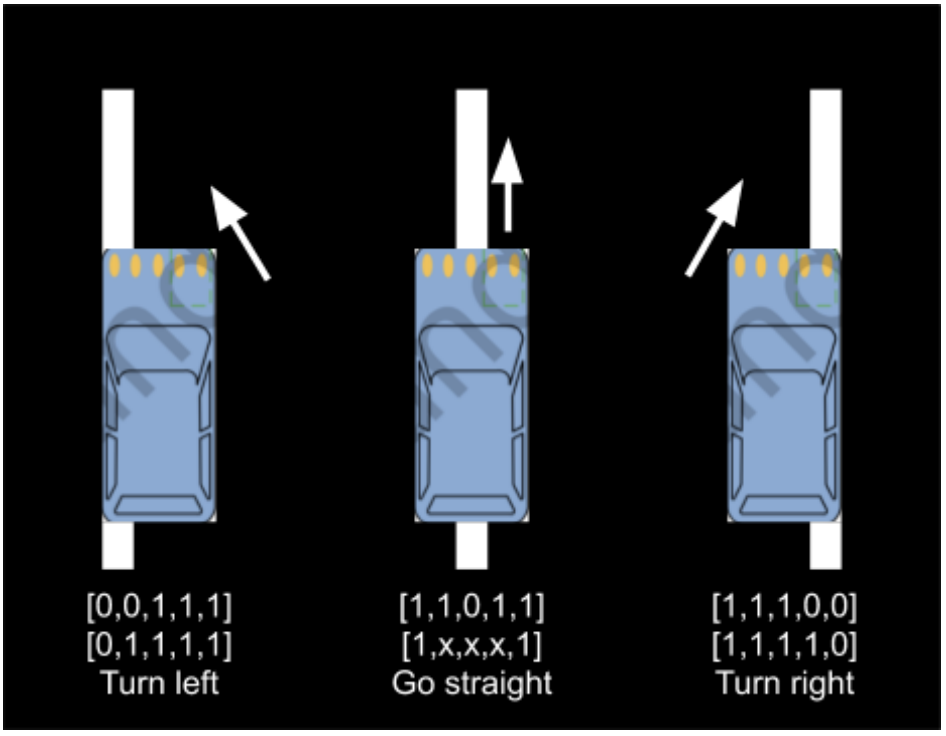
See if you can get it past multiple cones!

It's pretty difficult to get it to follow the path you want right? So why don't we try something to guide the car better - following a line!

Line Following

The Line following sensor has 5 probes. Each probe will return a 0 when it sees white and a 1 when it sees black. It returns this in a 'list' of the five probes. Here's some examples:

[0,0,0,0,0] - all sensors see white in this case
[0,0,1,1,1] - left two sensors see white, rest see black
[1,1,1,1,0] - most right sensor sees white, rest see black



So let's code to follow some lines.

1. We'll need the library for line following to the directory
   and then: *cp -r SunFounder_PiCar-S/example/SunFounder_Line_Follower/ .*
   You can use the tab key to autocomplete folder names. Don't miss the full stop at the end! You can check the folder has copied by typing *ls* and you should now see SunFounder_Line_Follower as one of the folders listed. Ask for help if you're unsure.

2. Start a new python script with *nano line.py* and make sure you have the following setup functions to the top of the file (check for which are in Caps):

```
import time
import picar
from SunFounder_Line_Follower import Line_Follower
fw = picar.front_wheels.Front_Wheels(db='config')
bw = picar.back_wheels.Back_Wheels(db='config')
lf = Line_Follower.Line_Follower()
```

3. Now let's try out some if statements, we'll need a *while True* so that your code will loop forever. The code which you want to repeat needs to be **indented** - use the tab key. 'lf_status' is how we read the line following sensor:

```
while True:
	lf_status = lf.read_digital()
	print(lf_status)
```

4. The ‘print’ in the code will display ‘lf_status’ on the command line for you to check. Let’s test this by typing *python line.py.* You should see the list of 5 sensors in the layout [x,x,x,x,x] showing on the screen. If you place the car on the track and move it by hand across the white line you should see the 0s move across the sensors.

5. Here’s some example code to get you started line following. (previous code in grey):

```
import time
import picar
from SunFounder_Line_Follower import Line_Follower
fw = picar.front_wheels.Front_Wheels(db='config')
bw = picar.back_wheels.Back_Wheels(db='config')
lf = Line_Follower.Line_Follower()

def main():
	while True:
		lf_status = lf.read_digital()
		print(lf_status)
		bw.speed = 20
		bw.forward()
		time.sleep(0.5)
		if lf_status == [0,0,1,1,1] or lf_status == [0,1,1,1,1]:
			fw.turn_left()
		elif lf_status == [1,1,1,0,0] or lf_status == [1,1,1,0,0]:
			fw.turn_right()
		else:
			fw.turn_straight()

if __name__ == '__main__':
	try:
		main()
	except KeyboardInterrupt:
		bw.stop()
		fw.turn_straight()
```

This uses a function we’ve called ‘main’ for the line following code and ‘except KeyboardInterrupt’ function so that when you stop the code with *ctrl+c* it will trigger to stop the motors spinning. Otherwise your car will keep driving forever!

It also uses ‘if statements’ - ‘if’ ‘elif’ ‘else’ what this is doing:

- **if** we see the line on the left turn left,
- **else if** we see the line on the right, turn right
- **else** turn straight

Now it’s down to you to improve the code and make the best line follower you can! Try using more accurate turning, more if statements, more back wheel speeds, time delays...