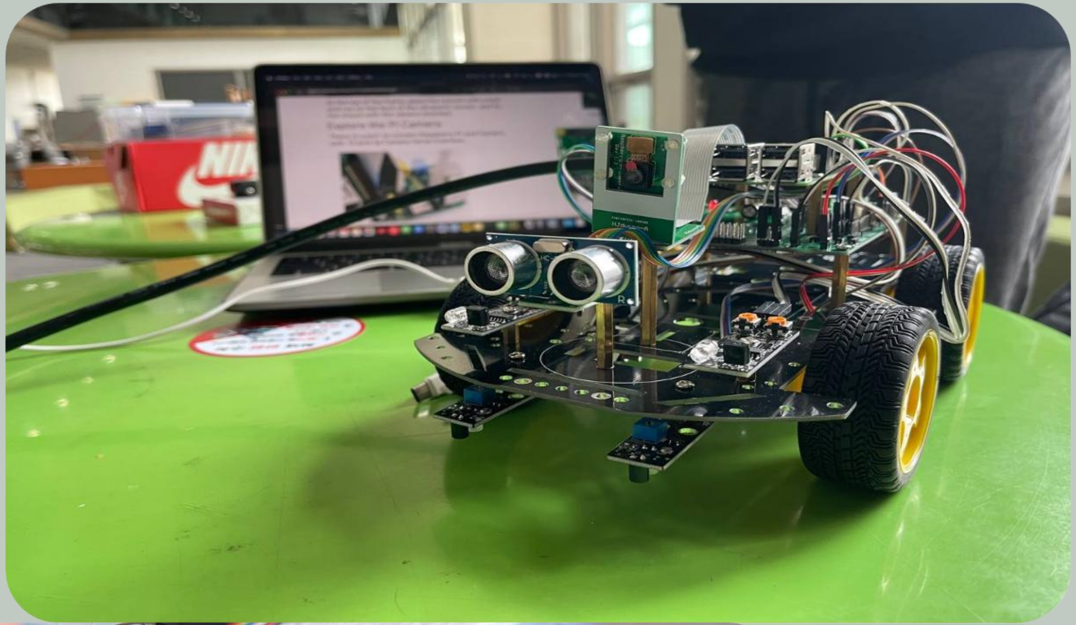
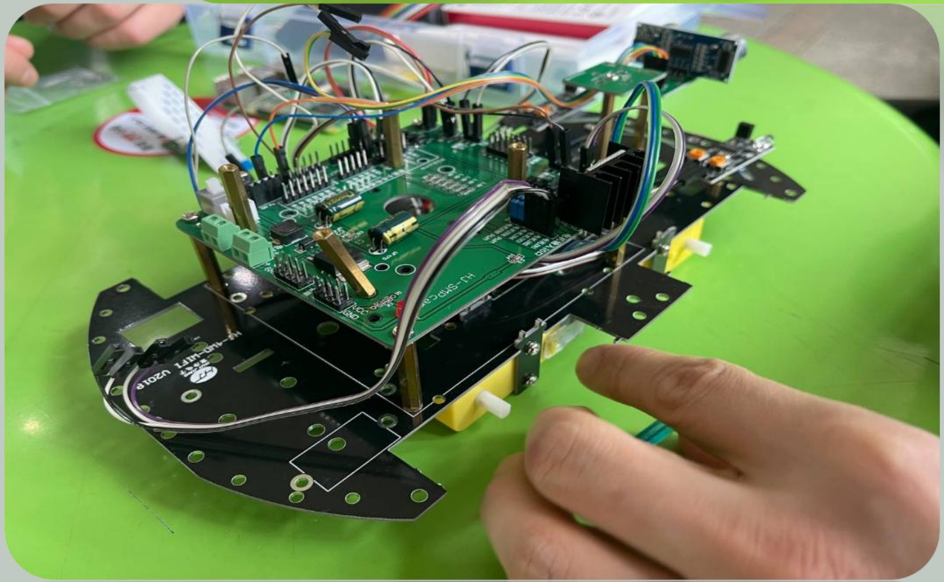
VIP PTOJECT





Team: “**isee**”

June 20, 2022

Project presentation

Group members

- 01 KHALILOV ASADBEK 12194882
- 06 AZODOV JAVOKHIR 12194924
- 03 YUSUPOV ELBEK 12194909
- 04 MIRZABAKHROMOV ULUGBEK 12194914
- 05 MAKHANNADJONOV IZZATULLOKH 12194921
- 06 KOMILOV KOBILJON 12194936

ABTRACT

01 BASIC KNOWLEDGE ABOUT RASPBERRY PI RC SMART CAR AND BASIC CONFIGURATION

02 THROUGH THIS SEMESTER, OUR COURSE MAINLY FOCUSED TO ASSEMBLE ALL PARTS INCLUDING EVERYTHING LIKE SENSOR, LIDAR AND ETC, AND WITH CONTROLLING THE VEHICLE WITH BASIC ALGORITHMS

03 HOW TO ASSEMBLE THE BODY PART INCLUDING THE SENSOR, DC MOTOR, CAMERA, MAIN BOARD AND ANOTHER IMPORTANT PARTS

RASPBERRY PI RC SMART CAR

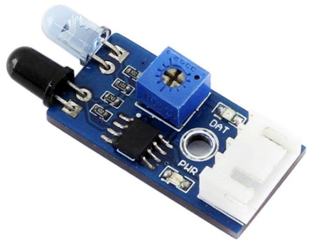
01

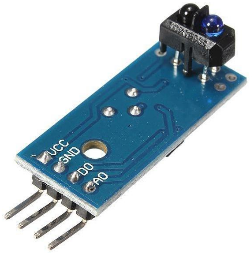
Raspberry Pi OS is a Debian-based operating system for Raspberry Pi. Since 2013, it has been officially provided by the Raspberry Pi Foundation as the primary operating system for the Raspberry Pi family of compact single-board computers. The first build was released on July 15, 2012. As the Raspberry Pi had no officially provided operating system at the time, the Raspberry Pi Foundation decided to build off of the work done by the Raspbian project and began producing and releasing their own version of the software.

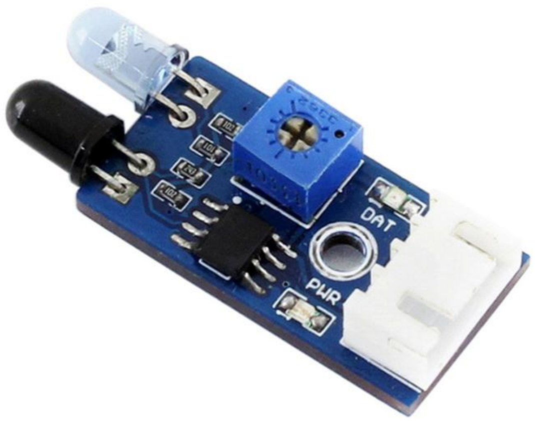
RASPBERRY PI RC SMART CAR

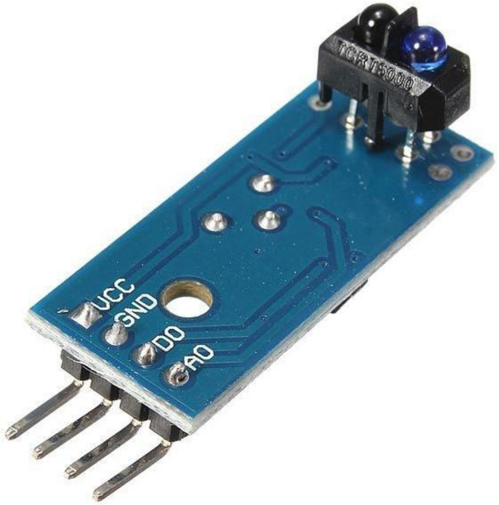
Sensors

Raspberry Pi









IR Sensor

By the turn on LED or any other electronic device whenever any obstacle comes nearer the IR sensor. Infrared Photodiode on the circuit and turn on power then Energy from InfraRed radiation is absorbed by the P-N junction of the InfraRed photodiode and is converted to electrical energy

Camera

Camera is an important part of a smart car. Canny Edge detection software is used for reducing the weight of videos but it preserves important information by detecting the objects edges.

Canny Edge detection workflow:

1.Initialize the images.

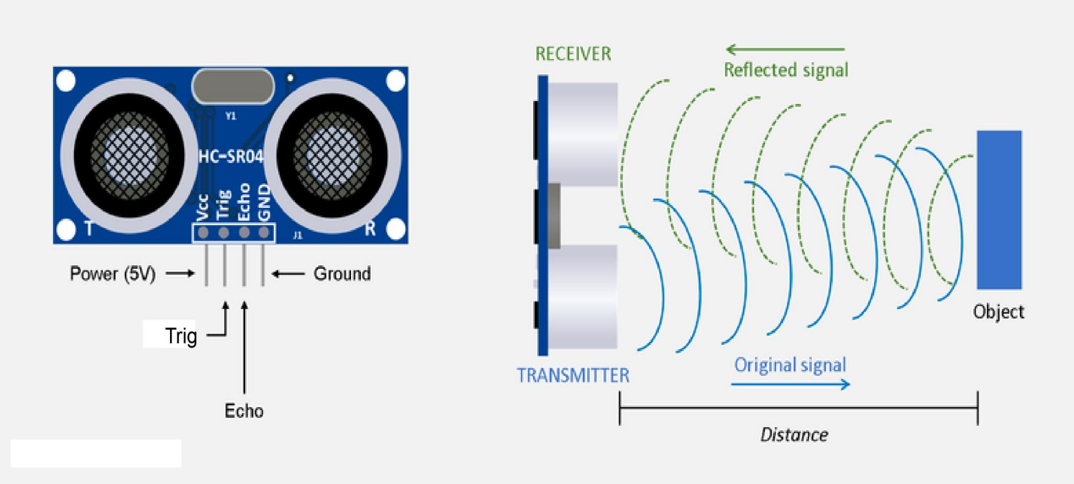
2.Initialize the raspberry cam by activating it.

3.Grab the current frame of the scene through RaspiCam.

4.Convert RGB to GrayScale image.

5.Apply Canny Edge Detector.

6.Display the results for every frame.



Ultrasonic Sensor

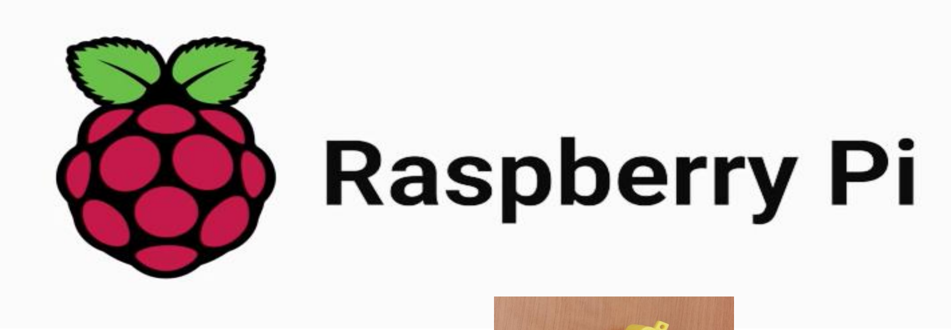
Ultrasonic sensors emit an acoustic weave between 20 hertz and 20 kilohertz and determine the distance. Ultrasonic sensors can be used in many fields. For instance, it can be used as parking assistance sensors in cars.

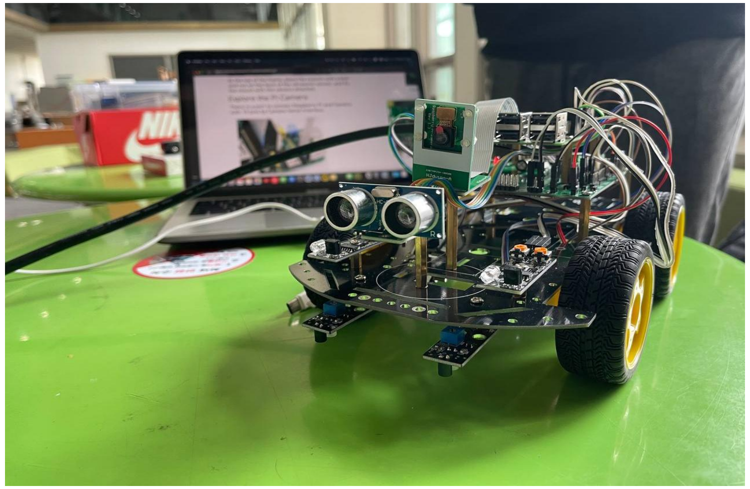
DC Motor

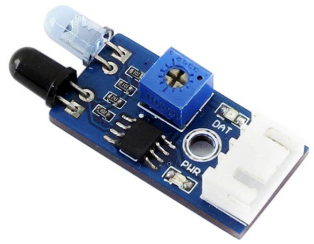
A DC motor or direct current motor is an electrical machine that transforms electrical energy into mechanical energy by creating a magnetic field that is powered by direct current. When a DC motor is powered, a magnetic field is created in its stator. The field attracts and repels magnets on the rotor; this causes the rotor to rotate. To keep the rotor continually rotating, the commutator that is attached to brushes connected to the power source supply current to the motors wire windings.

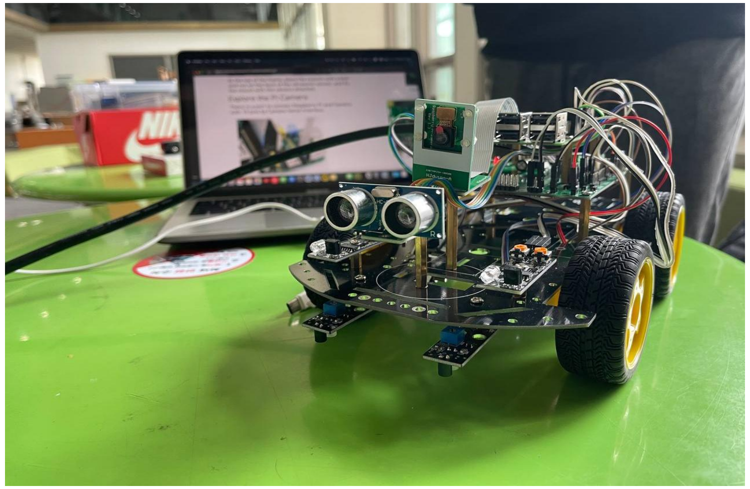
RASPBERRY PI RC SMART CAR

Challenges









Challenges:

There were several challenges during our classes. One of the DC motors became very weak and is causing imprecise maneuvers.

Moreover, the IR sensors did not always give the correct response to the obstacles. Instead of detecting an obstacle and moving around the obstacle smart car started acting randomly.

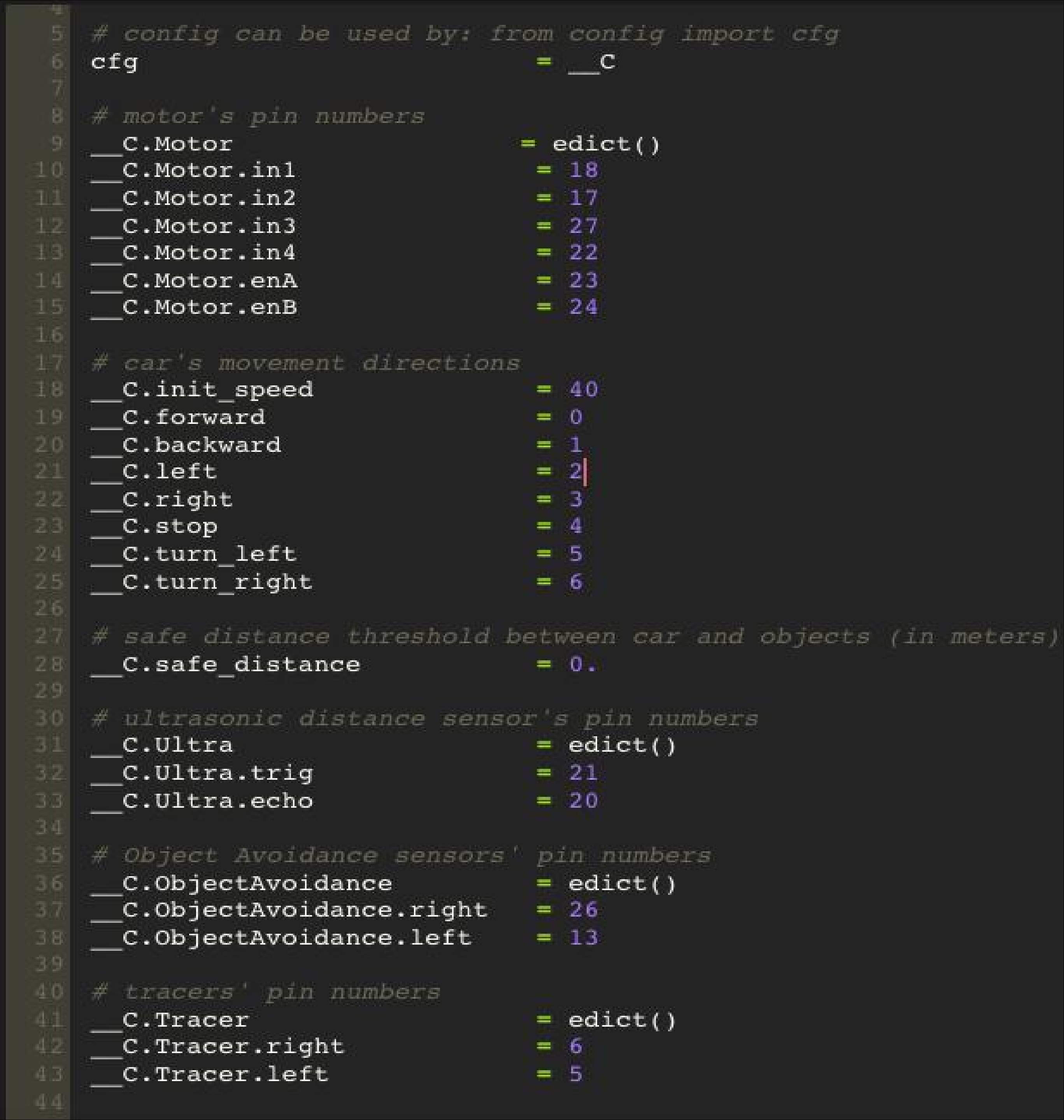
03

RASPERRY PI RC SMART CAR

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Originally developed by Intel, it was later supported by Willow Garage then Itseez. The library is cross-platform and free for use under the open-source Apache 2 License.

Configuration

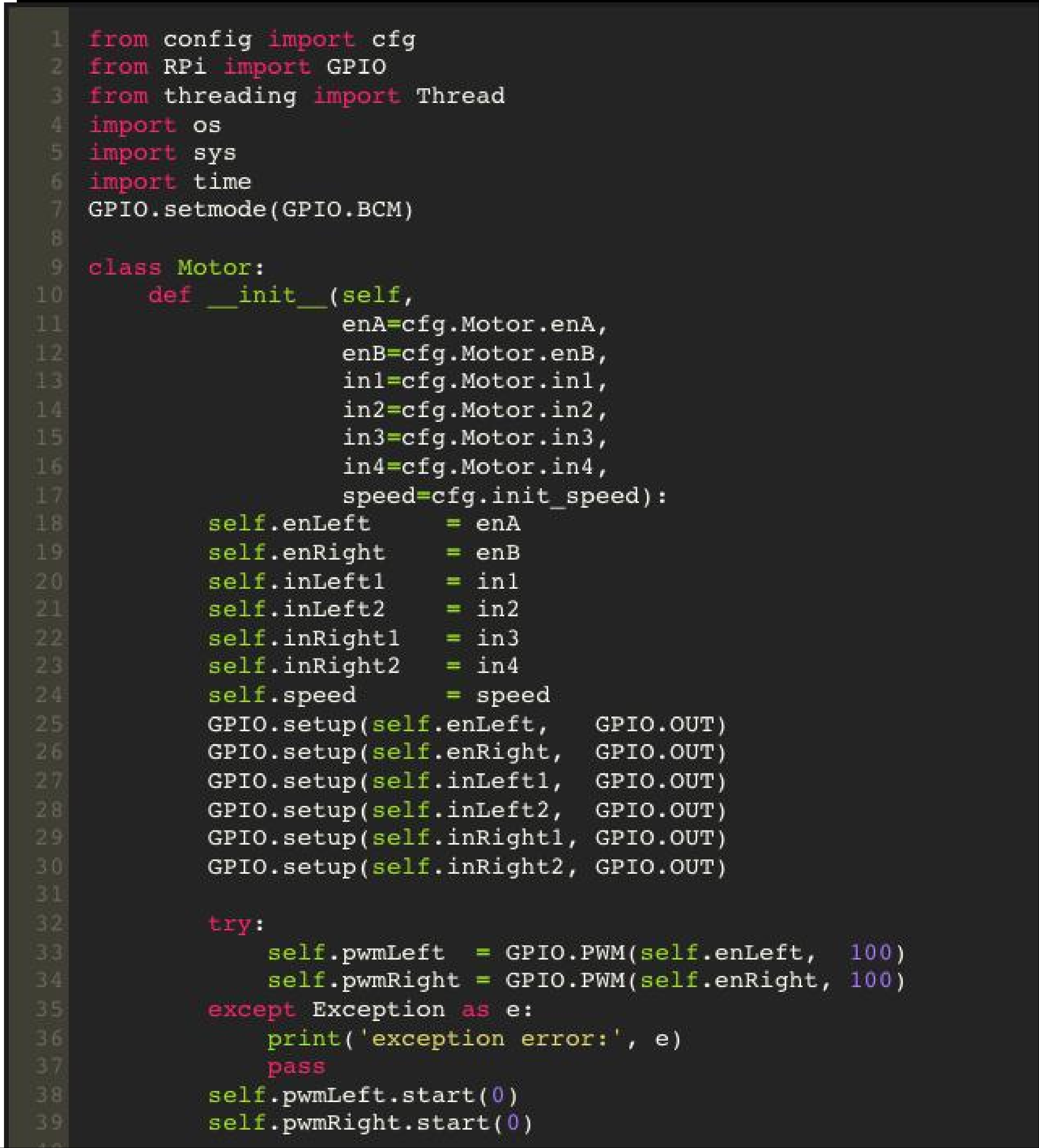
(PINs, distance, speed)



```
# config can be used by: from config import cfg
cfg                         = __C

# motor's pin numbers
__C.Motor                  = edict()
__C.Motor.in1               = 18
__C.Motor.in2               = 17
__C.Motor.in3               = 27
__C.Motor.in4               = 22
__C.Motor.enA               = 23
__C.Motor.enB               = 24

# car's movement directions
__C.init_speed              = 40
__C.forward                 = 0
__C.backward                = 1
__C.left                    = 2
__C.right                   = 3
__C.stop                    = 4
__C.turn_left               = 5
__C.turn_right              = 6

# safe distance threshold between car and objects (in meters)
__C.safe_distance           = 0.

# ultrasonic distance sensor's pin numbers
__C.Ultra                   = edict()
__C.Ultra.trig              = 21
__C.Ultra.echo              = 20

# Object Avoidance sensors' pin numbers
__C.ObjectAvoidance         = edict()
__C.ObjectAvoidance.right   = 26
__C.ObjectAvoidance.left    = 13

# tracers' pin numbers
__C.Tracer                  = edict()
__C.Tracer.right            = 6
__C.Tracer.left             = 5
```

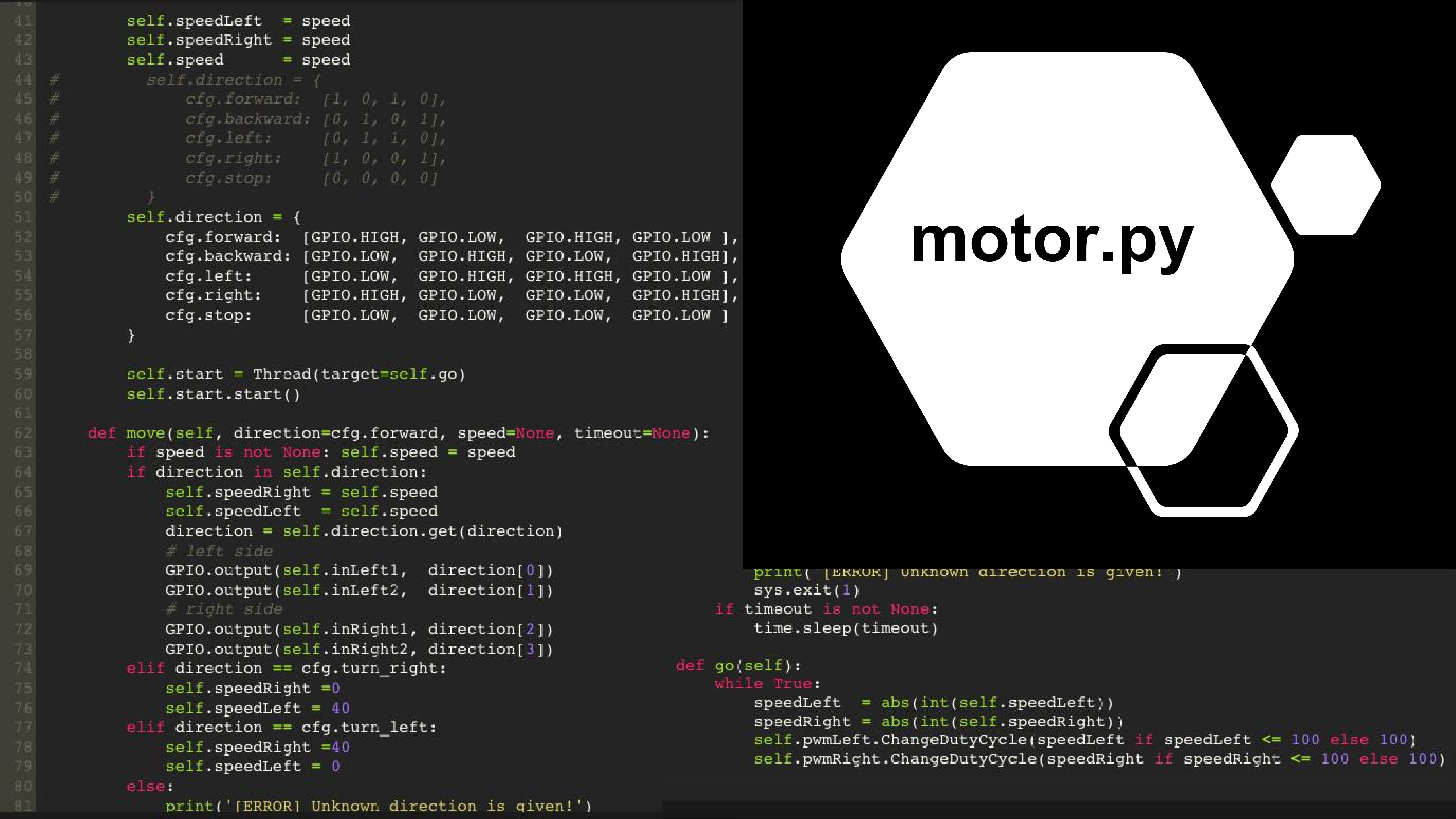
motor.py

(python code for running motor)



```python
from config import cfg
from RPi import GPIO
from threading import Thread
import os
import sys
import time
GPIO.setmode(GPIO.BCM)

class Motor:
    def __init__(self,
                 enA=cfg.Motor.enA,
                 enB=cfg.Motor.enB,
                 in1=cfg.Motor.in1,
                 in2=cfg.Motor.in2,
                 in3=cfg.Motor.in3,
                 in4=cfg.Motor.in4,
                 speed=cfg.init_speed):
        self.enLeft     = enA
        self.enRight    = enB
        self.inLeft1    = in1
        self.inLeft2    = in2
        self.inRight1   = in3
        self.inRight2   = in4
        self.speed      = speed
        GPIO.setup(self.enLeft,   GPIO.OUT)
        GPIO.setup(self.enRight,  GPIO.OUT)
        GPIO.setup(self.inLeft1,  GPIO.OUT)
        GPIO.setup(self.inLeft2,  GPIO.OUT)
        GPIO.setup(self.inRight1, GPIO.OUT)
        GPIO.setup(self.inRight2, GPIO.OUT)

        try:
            self.pwmLeft  = GPIO.PWM(self.enLeft,  100)
            self.pwmRight = GPIO.PWM(self.enRight, 100)
        except Exception as e:
            print('exception error:', e)
            pass
        self.pwmLeft.start(0)
        self.pwmRight.start(0)
```

motor.py

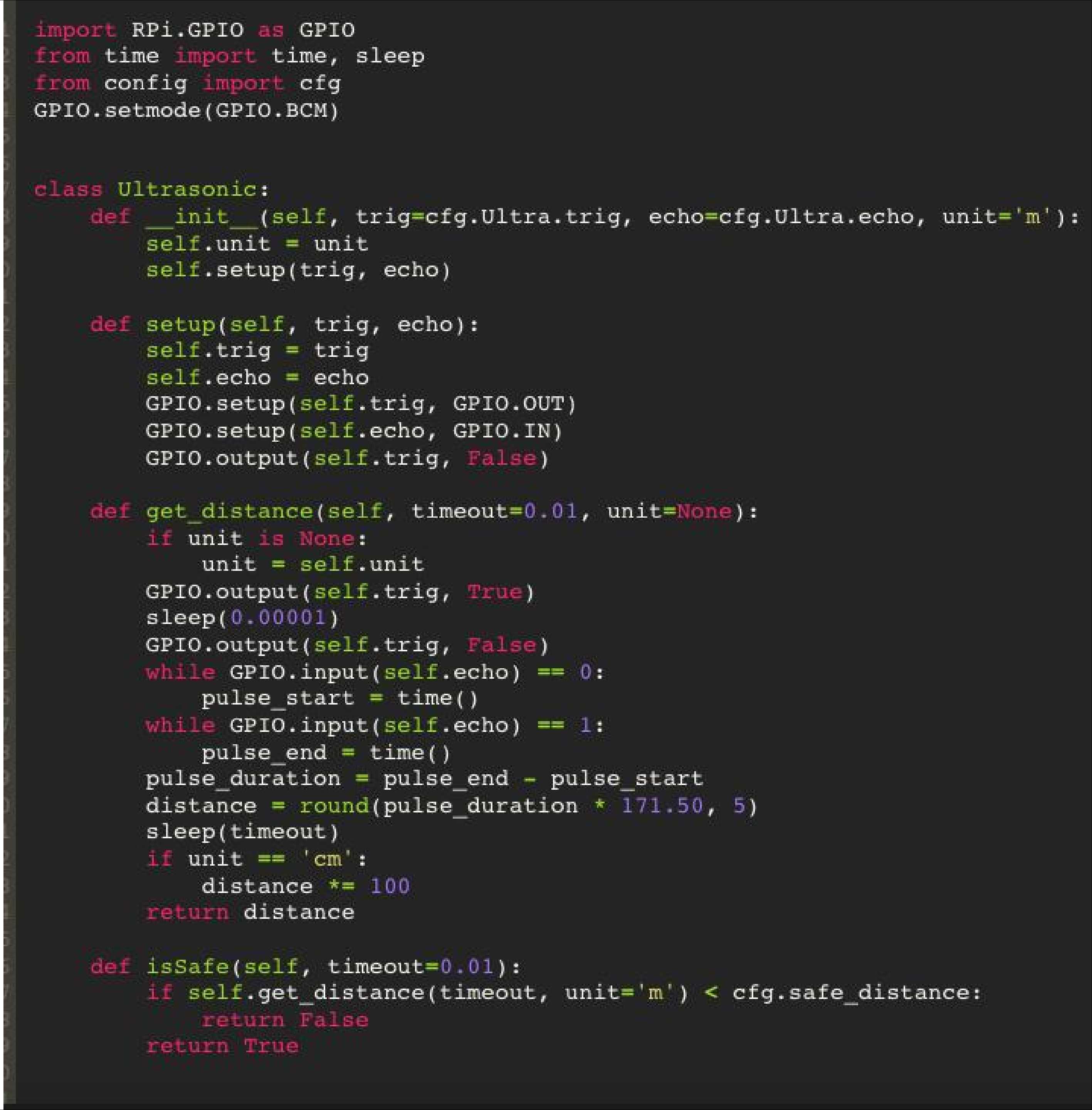


```
        self.speedLeft  = speed
        self.speedRight = speed
        self.speed      = speed
#         self.direction = {
#             cfg.forward:  [1, 0, 1, 0],
#             cfg.backward: [0, 1, 0, 1],
#             cfg.left:     [0, 1, 1, 0],
#             cfg.right:    [1, 0, 0, 1],
#             cfg.stop:     [0, 0, 0, 0]
#         }
        self.direction = {
            cfg.forward:  [GPIO.HIGH, GPIO.LOW,  GPIO.HIGH, GPIO.LOW ],
            cfg.backward: [GPIO.LOW,  GPIO.HIGH, GPIO.LOW,  GPIO.HIGH],
            cfg.left:     [GPIO.LOW,  GPIO.HIGH, GPIO.HIGH, GPIO.LOW ],
            cfg.right:    [GPIO.HIGH, GPIO.LOW,  GPIO.LOW,  GPIO.HIGH],
            cfg.stop:     [GPIO.LOW,  GPIO.LOW,  GPIO.LOW,  GPIO.LOW ]
        }

        self.start = Thread(target=self.go)
        self.start.start()

    def move(self, direction=cfg.forward, speed=None, timeout=None):
        if speed is not None: self.speed = speed
        if direction in self.direction:
            self.speedRight = self.speed
            self.speedLeft  = self.speed
            direction = self.direction.get(direction)
            # left side
            GPIO.output(self.inLeft1,  direction[0])
            GPIO.output(self.inLeft2,  direction[1])
            # right side
            GPIO.output(self.inRight1, direction[2])
            GPIO.output(self.inRight2, direction[3])
        elif direction == cfg.turn_right:
            self.speedRight =0
            self.speedLeft = 40
        elif direction == cfg.turn_left:
            self.speedRight =40
            self.speedLeft = 0
        else:
            print('[ERROR] Unknown direction is given!')
```

```
            print('[ERROR] Unknown direction is given!')
            sys.exit(1)
        if timeout is not None:
            time.sleep(timeout)

    def go(self):
        while True:
            speedLeft  = abs(int(self.speedLeft))
            speedRight = abs(int(self.speedRight))
            self.pwmLeft.ChangeDutyCycle(speedLeft if speedLeft <= 100 else 100)
            self.pwmRight.ChangeDutyCycle(speedRight if speedRight <= 100 else 100)
```

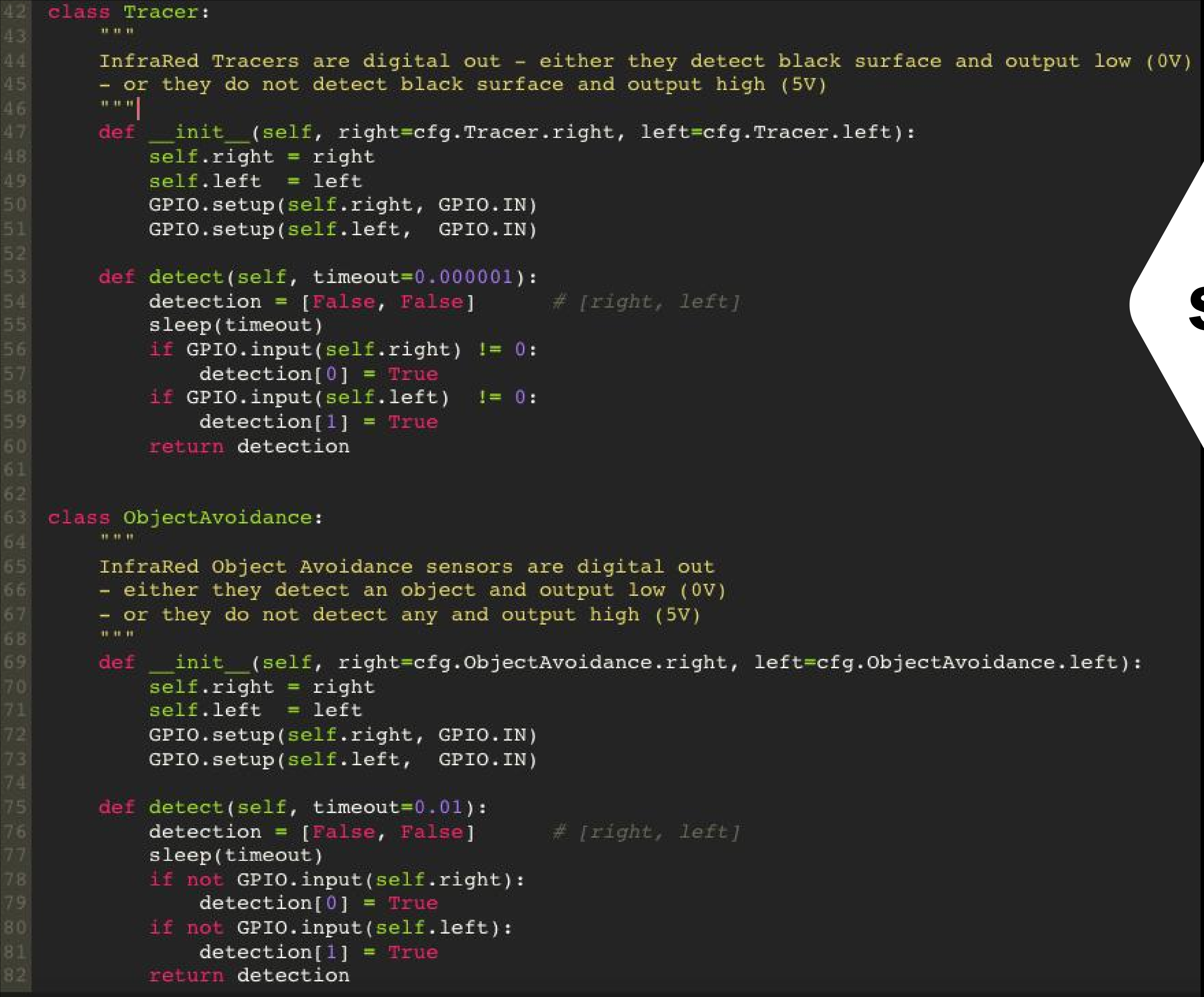
sensor.py

(python code for sensor configurations)



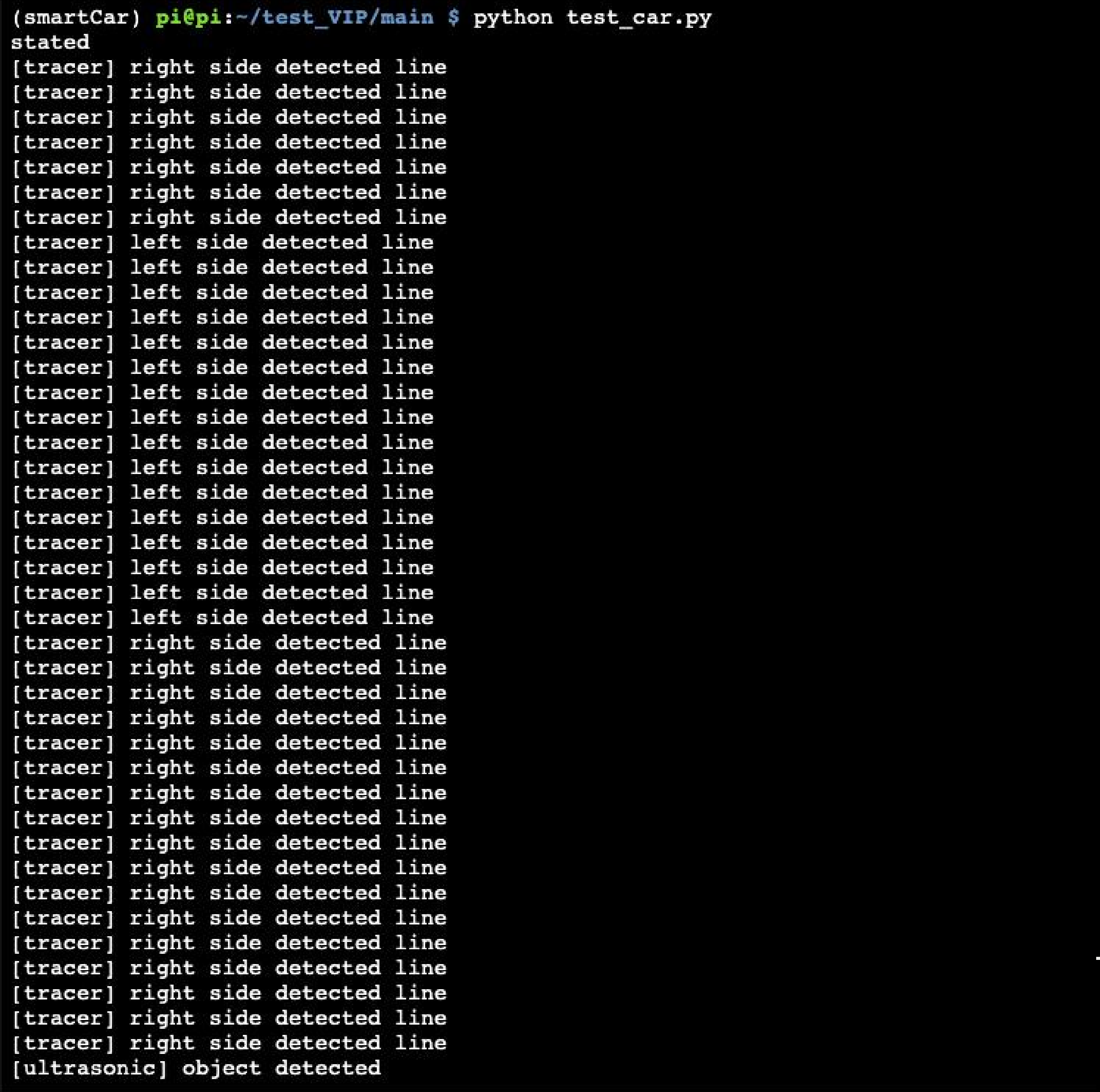
```python
import RPi.GPIO as GPIO
from time import time, sleep
from config import cfg
GPIO.setmode(GPIO.BCM)


class Ultrasonic:
    def __init__(self, trig=cfg.Ultra.trig, echo=cfg.Ultra.echo, unit='m'):
        self.unit = unit
        self.setup(trig, echo)

    def setup(self, trig, echo):
        self.trig = trig
        self.echo = echo
        GPIO.setup(self.trig, GPIO.OUT)
        GPIO.setup(self.echo, GPIO.IN)
        GPIO.output(self.trig, False)

    def get_distance(self, timeout=0.01, unit=None):
        if unit is None:
            unit = self.unit
        GPIO.output(self.trig, True)
        sleep(0.00001)
        GPIO.output(self.trig, False)
        while GPIO.input(self.echo) == 0:
            pulse_start = time()
        while GPIO.input(self.echo) == 1:
            pulse_end = time()
        pulse_duration = pulse_end - pulse_start
        distance = round(pulse_duration * 171.50, 5)
        sleep(timeout)
        if unit == 'cm':
            distance *= 100
        return distance

    def isSafe(self, timeout=0.01):
        if self.get_distance(timeout, unit='m') < cfg.safe_distance:
            return False
        return True
```

sensor.py



```python
class Tracer:
    """
    InfraRed Tracers are digital out - either they detect black surface and output low (0V)
    - or they do not detect black surface and output high (5V)
    """
    def __init__(self, right=cfg.Tracer.right, left=cfg.Tracer.left):
        self.right = right
        self.left  = left
        GPIO.setup(self.right, GPIO.IN)
        GPIO.setup(self.left,  GPIO.IN)

    def detect(self, timeout=0.000001):
        detection = [False, False]      # [right, left]
        sleep(timeout)
        if GPIO.input(self.right) != 0:
            detection[0] = True
        if GPIO.input(self.left)  != 0:
            detection[1] = True
        return detection


class ObjectAvoidance:
    """
    InfraRed Object Avoidance sensors are digital out
    - either they detect an object and output low (0V)
    - or they do not detect any and output high (5V)
    """
    def __init__(self, right=cfg.ObjectAvoidance.right, left=cfg.ObjectAvoidance.left):
        self.right = right
        self.left  = left
        GPIO.setup(self.right, GPIO.IN)
        GPIO.setup(self.left,  GPIO.IN)

    def detect(self, timeout=0.01):
        detection = [False, False]      # [right, left]
        sleep(timeout)
        if not GPIO.input(self.right):
            detection[0] = True
        if not GPIO.input(self.left):
            detection[1] = True
        return detection
```

OUTPUT



```
(smartCar) pi@pi:~/test_VIP/main $ python test_car.py
stated
[tracer] right side detected line
[tracer] right side detected line
[tracer] right side detected line
[tracer] right side detected line
[tracer] right side detected line
[tracer] right side detected line
[tracer] right side detected line
[tracer] left side detected line
[tracer] left side detected line
[tracer] left side detected line
[tracer] left side detected line
[tracer] left side detected line
[tracer] left side detected line
[tracer] left side detected line
[tracer] left side detected line
[tracer] left side detected line
[tracer] left side detected line
[tracer] left side detected line
[tracer] left side detected line
[tracer] left side detected line
[tracer] left side detected line
[tracer] left side detected line
[tracer] left side detected line
[tracer] right side detected line
[tracer] right side detected line
[tracer] right side detected line
[tracer] right side detected line
[tracer] right side detected line
[tracer] right side detected line
[tracer] right side detected line
[tracer] right side detected line
[tracer] right side detected line
[tracer] right side detected line
[tracer] right side detected line
[tracer] right side detected line
[tracer] right side detected line
[tracer] right side detected line
[tracer] right side detected line
[tracer] right side detected line
[tracer] right side detected line
[ultrasonic] object detected
```

THANK YOU