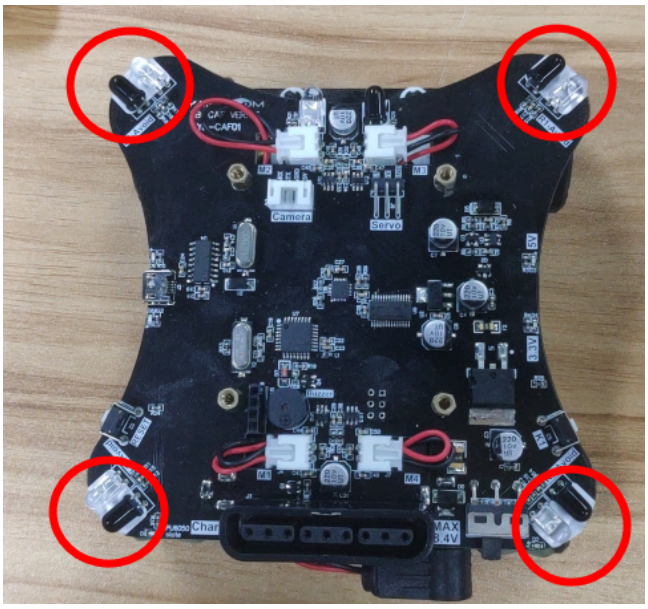
6.8 Car Translation mode

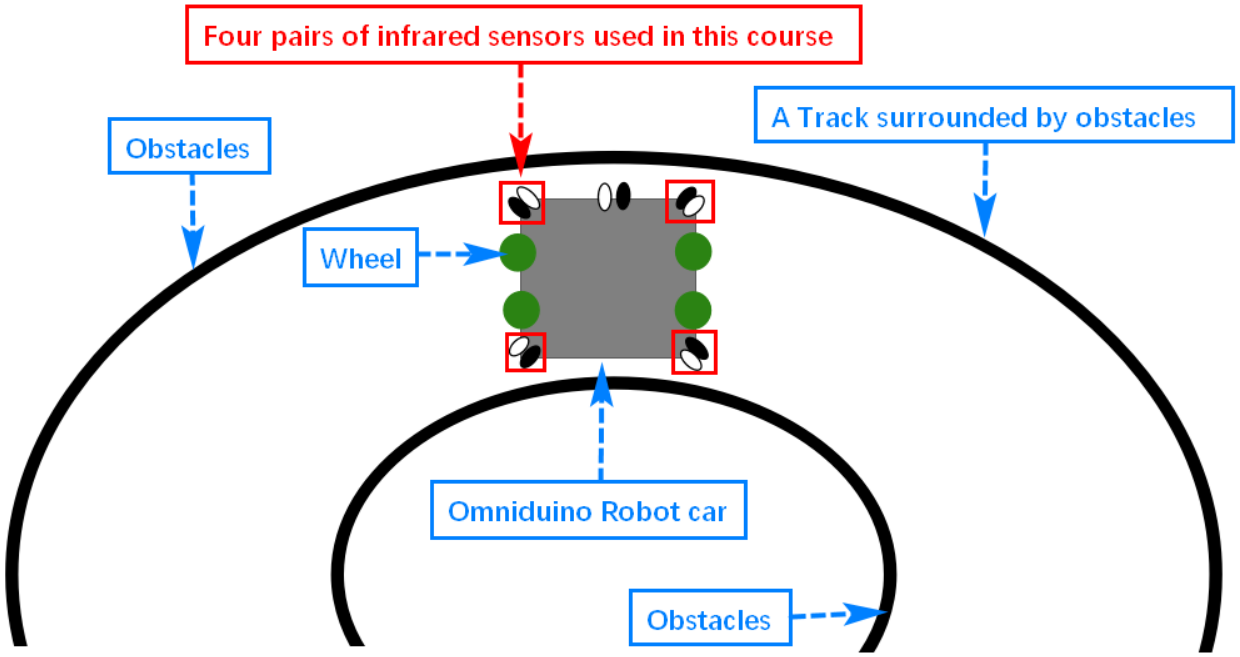
1. Learning goal:

As shown in the figure below, the red wire frame contains the four pairs of infrared sensors used in this course.

According to the distance between the four corners of the infrared sensor and the obstacle, keep the distance between the body and the obstacle, and let the car move along the road.



As shown below:



2. Experimental phenomena:

Turn on the power switch of the car. Put the car on the road with a wall on both sides, press the K1 button, the car will whistle once, then it will remain in the middle of the road and left

translation along the road.

3. About infrared sensor

This course mainly uses infrared obstacle avoidance sensors(infrared pair tubes), white is sent tube, black is receive tube.
When an obstacle is detected, the closer the obstacle is to the robot car, the smaller the analog value returned by the infrared sensor.

4. Code analysis

4.1 The main function of the carLateral() is to detect the analog value returned by the sensor, input the analog value into the PID calculation, and obtain the corresponding wheel deceleration or acceleration, which can adjust the speed of the car motor and keeping the car in the middle of the road as much as possible. The parameter speed is the initial speed, the direction: 0 is to the left, and 1 is to the right.

```
void carLateral(int speed, int direction)
{
  int speed_L1 = speed;
  int speed_L2 = speed;
  int speed_R1 = speed;
  int speed_R2 = speed;

  ir_L1 = analogRead(IR_SENSOR_L1) / 10;
  ir_L2 = analogRead(IR_SENSOR_L2) / 10;
  ir_R1 = analogRead(IR_SENSOR_R1) / 10;
  ir_R2 = analogRead(IR_SENSOR_R2) / 10;

  int pid_l1 = PIDCal_IR(PID_IR, ir_L1);
  int pid_l2 = PIDCal_IR(PID_IR, ir_L2);
  int pid_r1 = PIDCal_IR(PID_IR, ir_R1);
  int pid_r2 = PIDCal_IR(PID_IR, ir_R2);

  if (direction == 0)            // left
  {
    speed_L1 = speed_L1 + pid_l2;
    speed_L2 = speed_L2 + pid_l1;
    speed_R1 = speed_R1 + pid_r1;
    speed_R2 = speed_R2 + pid_r2;

    carRun(-speed_L1, speed_R1, speed_L2, -speed_R2);
  }
  else if (direction == 1)      // right
  {
    speed_L1 = speed_L1 + pid_l1;
    speed_L2 = speed_L2 + pid_l2;
    speed_R1 = speed_R1 + pid_r2;
    speed_R2 = speed_R2 + pid_r1;

    carRun(speed_L1, -speed_R1, -speed_L2, speed_R2);
  }
}
```

4.2 Increase the PID calculation PID_IR, adjust the speed of each motor of the car, make the

car keep running in the middle of the road as much as possible.
And the turn function also uses this PID calculation to adjust the motor speed to provide the centripetal force required for cornering.

```
typedef struct
{
  float SetPoint;    //Desired value
  float Proportion; // Proportional Const
  float Integral;    // Integral Const
  float Derivative; // Derivative Const
  float LastError;  // Error[-1]
  float PrevError;  // Error[-2]
  float SumError;    // Sums of Errors
} PID;

/*==============================================
PID Calculation part
==============================================*/
PID PID_MID = {60, 0.5, 0, 0, 0, 0, 0};
PID PID_IR = {40, 1.2, 0, 0, 0, 0, 0};
float PIDCal_IR(PID pid, float nowValue);
```

```
float PIDCal_IR(PID pid, float nowValue)
{
  float dError, Error;
  Error = pid.SetPoint - nowValue;
  pid.SumError += Error;
  dError = pid.LastError - pid.PrevError;
  pid.PrevError = pid.LastError;
  pid.LastError = Error;

  double val_sensor = pid.Proportion * Error
            + pid.Integral * pid.SumError
            + pid.Derivative * dError;
  return -val_sensor;
}
```

4.3 In the loop function scan button, when the button status is true(press K1 button for the first time), the buzzer whistle once, the car translation mode is started. When the button state is false(press K1 button for the second time), the buzzer whistle once, the car exit translation mode.

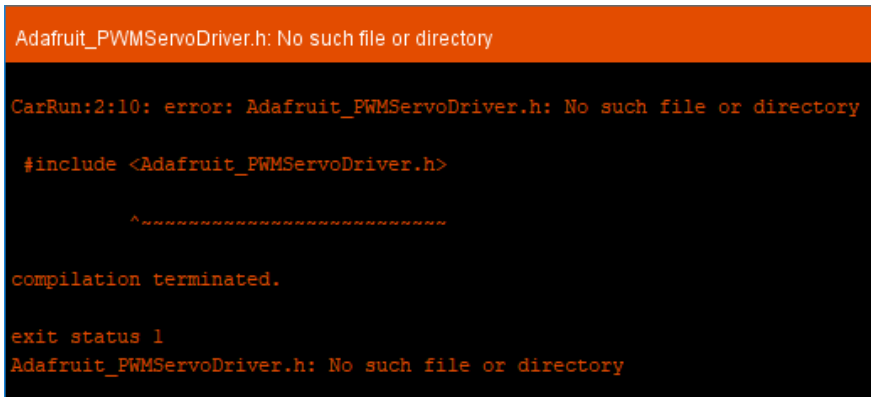
```
void loop()
{
  keyscan();
  if (button_press)
  {
    ir_rgb();
    carLateral(CarSpeedControl, 0);
  }
  else
  {
    brake();
    clearRGB();
  }
}
```

5. Compiling and downloading code

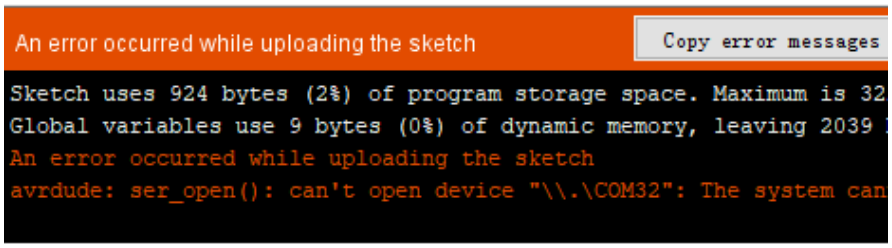
5.1 Open the **Surround.ino** program , select the serial port and click upload directly (the Omniduino car must first be connected to the computer via the USB data cable).

5.2 If there is an error like the following, it means that the library file is missing. Please copy the library file provided by the Omniduino omnibus to the library file directory compiled by arduinoIDE.

please refer to 【3.Development Environment Construction】----【3.4 Add additional library files】



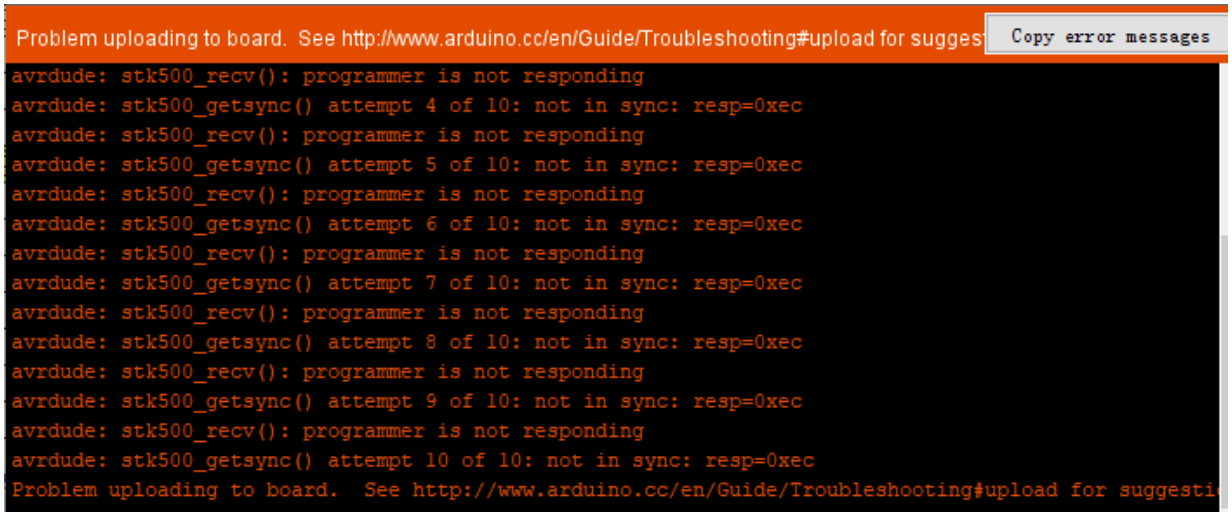
5.3 If the compilation passes normally, but the following error occurs during uploading, the reason may be that the wrong serial port or the serial port is occupied.



Solution: Open the device manager to see if there is a serial port with CH340 tag. If not, please restart the Omniduino car, then, re-plug the USB cable or replace a USB cable;
If there is a serial port number, we need to close the other serial port or assistant software, avoid serial port occupation, and then re-select the serial port to ArduinoIDE【Tool】-->【Port】.

5.4 After clicking the upload button, the upload is always displayed, but it can't be uploaded

successfully for a long time.



Because the uploading program and the WIFI camera communication is realized through the serial port, when the serial port is occupied by the WIFI camera, and the program cannot be uploaded.

Solution:

①Unplug the USB cable, turn off the power of the car, wait for the D2 indicator to go out.

②Then, plug in the USB data cable. At this time, your mobile phone should not connect the WiFi signal of the car.

③You can upload the program to the car according to the normal steps.

④After the program is successfully uploaded, unplug the USB data cable, open the power switch of the car. The corresponding experimental phenomenon will appear.

(Tip: If you upload APP control program. After the program is successfully uploaded, unplug the USB data cable, open the power switch of the car. Mobile phone connect the car to the WIFI signal, and then open the APP to control.)