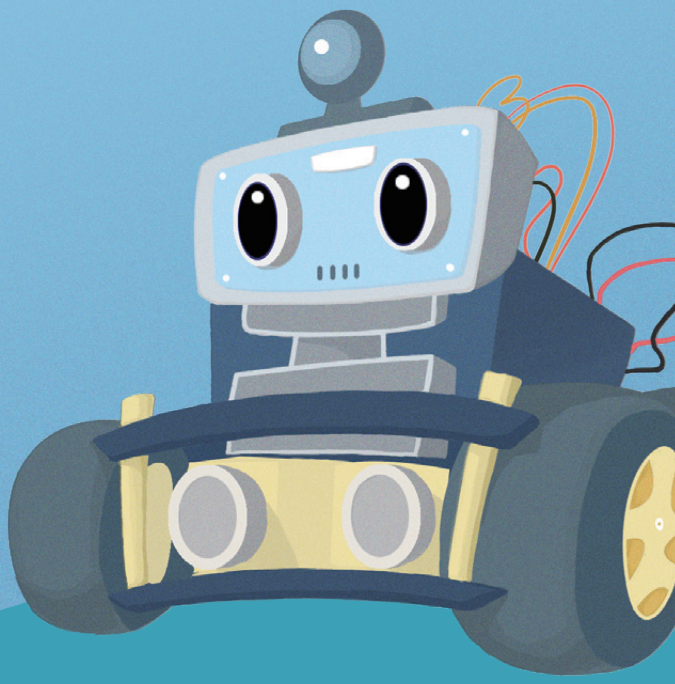
ELEGOO

3

SMART ROBOT
CAR V4.0 WITH CAMERA



Tracking Control

Introduction:

- In this lesson, we will teach you how to achieve the line-tracking function of the Smart Robot Car V 4.0 and make it move according to the black line we drawn.

If you want to make your own runway, here are some tips for you.

Tips:

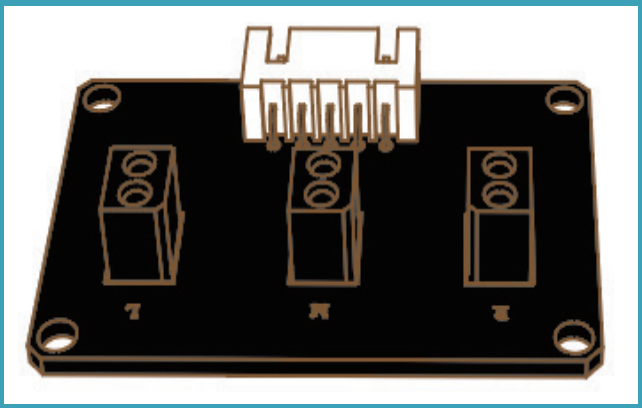
- The cured part of the line should be as smooth as possible. If the radius of the turning circle is too small, the car may probably miss the runway.

- In addition to line tracking, we can also expand our imagination to develop other program according to the theoretical of line tracking. For example, we can develop a program that confine the car to an area regardless of its movement.

Preparation:

- A Smart Robot Car (with battery)
  A USB Cable

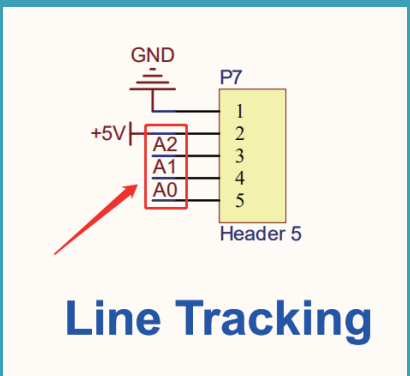
In our kit, the line-tracking function is achieved by the tracking module, and the tracking module is consisted of three photoelectric sensor and other electric components.



Please open the folder of last level for details: Related chip information -> ITR20001 and SmartRobot-Shield

The photoelectric sensor is composed of an infrared pair tube (transmitting/receiving) and an NPN silicon phototransistor, which is equipped with a light source and an optical receiving device. The light emitted by the light source is received by the photosensitive element through the reflection of the object to be measured, and then the required information is obtained through the processing of relevant circuits. It can be used to detect the change of ground light and shade and color, as well as detect the presence or absence of close objects.

Finally, let' s see which pin the three photoelectric sensors connect to and then we can start writing program.



✚ Please open Demo1 in the current folder.

✚ First of all, let' s define the related pins and variables:

```
C:/ //in DeviceDriverSet_xxx0.h
    /*ITR20001 Detection*/
    class DeviceDriverSet_ITR20001
    {
    public:
     bool DeviceDriverSet_ITR20001_Init(void);
     float DeviceDriverSet_ITR20001_getAnalogueexxx_L(void);
     float DeviceDriverSet_ITR20001_getAnalogueexxx_M(void);
     float DeviceDriverSet_ITR20001_getAnalogueexxx_R(void);
    #if _Test_DeviceDriverSet
     void DeviceDriverSet_ITR20001_Test(void);
    #endif

    private:
    #define PIN_ITR20001xxxL A2
    #define PIN_ITR20001xxxM A1
    #define PIN_ITR20001xxxR A0
    };
```

✚ We need to initial the pin first before applying and set the three pins as input mode respectively.

```
C:/ //in DeviceDriverSet_xxx0.cpp
    bool DeviceDriverSet_ITR20001::DeviceDriverSet_ITR20001_Init(void)
    {
     pinMode(PIN_ITR20001xxxL, INPUT);
     pinMode(PIN_ITR20001xxxM, INPUT);
     pinMode(PIN_ITR20001xxxR, INPUT);
     return false;
    }
```

And then put it into setup() for call.

```
C:/ void setup() {
      Serial.begin(9600);
      AppITR20001.DeviceDriverSet_ITR20001_Init();
    }
```

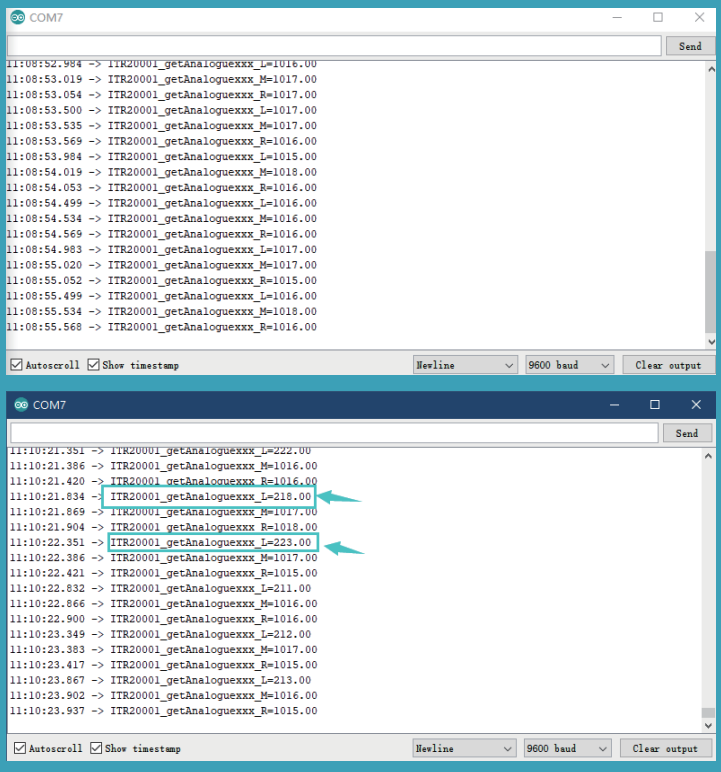
Finally, the data of the photoelectric sensor under the change of the external environment is obtained by reading the analog of the three pins.

```
C:/ //in DeviceDriverSet_xxx0.cpp
    float DeviceDriverSet_ITR20001::
    DeviceDriverSet_ITR20001_getAnalogueexxx_L(void)
    {
      return analogRead(PIN_ITR20001xxxL);
    }
    float DeviceDriverSet_ITR20001::
    DeviceDriverSet_ITR20001_getAnalogueexxx_M(void)
    {
      return analogRead(PIN_ITR20001xxxM);
    }
    float DeviceDriverSet_ITR20001::
    DeviceDriverSet_ITR20001_getAnalogueexxx_R(void)
    {
      return analogRead(PIN_ITR20001xxxR);
    }
```

✚ And then print it out in loop().

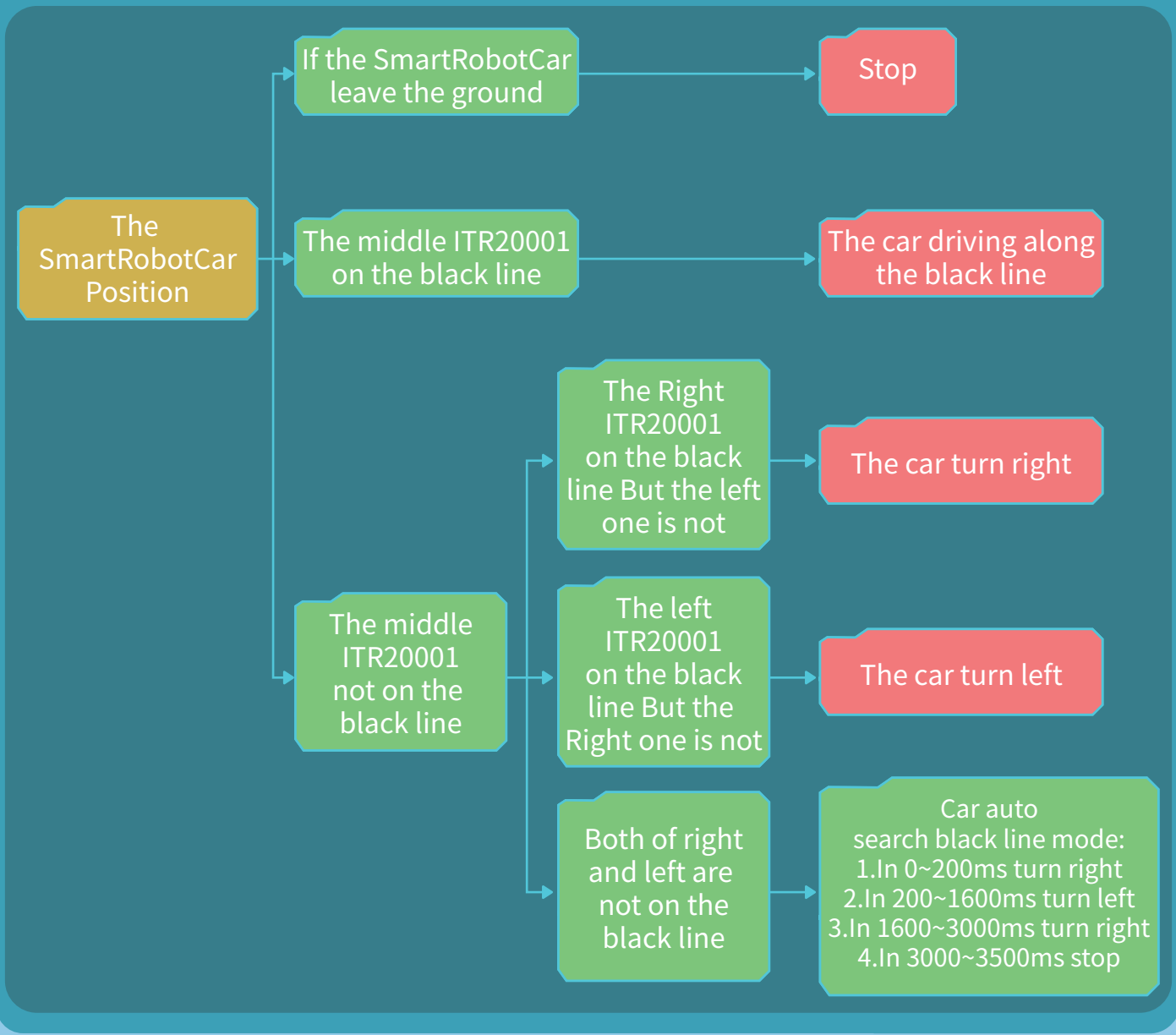
```
C:/ void loop() {
      static unsigned long print_time = 0;
      if (millis() - print_time > 500)
      {
        print_time = millis();
        Serial.print("ITR20001_getAnaloguexxx_L=");
        Serial.println
        (AppITR20001.DeviceDriverSet_ITR20001_getAnaloguexxx_L());
        Serial.print("ITR20001_getAnaloguexxx_M=");
        Serial.println
        (AppITR20001.DeviceDriverSet_ITR20001_getAnaloguexxx_M());
        Serial.print("ITR20001_getAnaloguexxx_R=");
        Serial.println
        (AppITR20001.DeviceDriverSet_ITR20001_getAnaloguexxx_R());
      }
    }
```

✚ Upload the program. (Please toggle the “Upload-Cam” button to “Upload” when uploading the program.) After the program has been uploaded successfully, please open the serial port and you will see the data of the three photoelectric sensors fluctuates around 1000. And if you block one of the photoelectric sensors with your hands, the corresponding detected value will be greatly reduced.



After learning the previous courses, we have mastered the use of photoelectric sensor. Next, we are going to achieve the tracking function by combining the car' s movement function learned in the lesson 2 and the photoelectric sensor.

At the beginning, let' s look at the whole flow chart of the tracking mode.



- Please open Demo2 in the current folder:
- First of all, let' s take a look at the relative definition of tracking function.

C:/ //in ApplicationFunctionSet_xxx0.h

```
class ApplicationFunctionSet
{
public:
  void ApplicationFunctionSet_Init(void);
  void ApplicationFunctionSet_Tracking(void);
  void ApplicationFunctionSet_SensorDataUpdate(void);
  void ApplicationFunctionSet_SerialPortDataAnalysis(void);
private:
  volatile float TrackingData_L;
  volatile float TrackingData_M;
  volatile float TrackingData_R;

  boolean TrackingDetectionStatus_R = false;
  boolean TrackingDetectionStatus_M = false;
  boolean TrackingDetectionStatus_L = false;

public:
  boolean Car_LeaveTheGround = true;

public:
//If the value read by the photoelectric sensor is within 250 ~ 850,
//the photoelectric sensor is in the black line.
  uint16_t TrackingDetection_S = 250;
  uint16_t TrackingDetection_E = 850;

//This is the minimum value obtained by the photoelectric sensor
//if the car leaves the ground
  uint16_t TrackingDetection_V = 950;
};
```

Then, we need to program a function to judge the range.

C:/ //in ApplicationFunctionSet_xxx0.cpp

```
static boolean
function_xxx(long x, long s, long e) //f(x)
{
 if (s <= x && x <= e)
  return true;
 else
  return false;
}
```

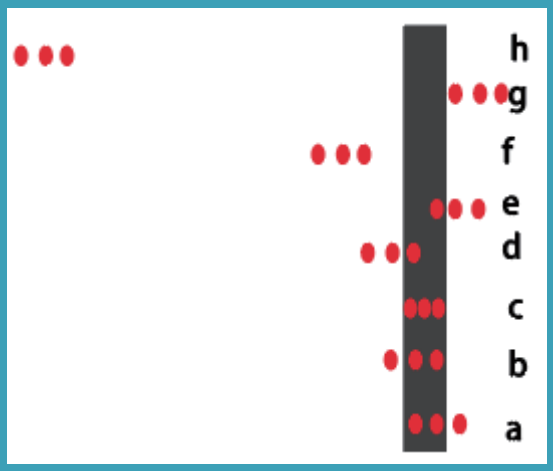
Then, we need to program the situation when the car is picked up.

C:/ //in ApplicationFunctionSet_xxx0.cpp

```
static bool ApplicationFunctionSet_SmartRobotCarLeaveTheGround(void)
{
 if (AppITR20001.DeviceDriverSet_ITR20001_getAnalogueexxx_R() >
    Application_FunctionSet.TrackingDetection_V &&
    AppITR20001.DeviceDriverSet_ITR20001_getAnalogueexxx_M() >
    Application_FunctionSet.TrackingDetection_V &&
    AppITR20001.DeviceDriverSet_ITR20001_getAnalogueexxx_L() >
    Application_FunctionSet.TrackingDetection_V)
 {
  Application_FunctionSet.Car_LeaveTheGround = false;
  return false;
 }
 else
 {
  Application_FunctionSet.Car_LeaveTheGround = true;
  return true;
 }
}
```

C:/
```
//in ApplicationFunctionSet_xxx0.cpp
void ApplicationFunctionSet::ApplicationFunctionSet_Tracking(void)
{
.....
 if (Application_SmartRobotCarxxx0.Functional_Mode == TraceBased_mode)
 {
  if (Car_LeaveTheGround == false)    {
   ApplicationFunctionSet_SmartRobotCarMotionControl(stop_it, 0);
   return;
}
.....
}
```

✚The preparatory work was completed, and then let’ s start writing the tracking function. Let's analyze the three ITR20001 online scenarios.



✚ a, b, c: When the middle ITR20001 is on the black line and the black line can be detected, the car will keep going straight.

C:/
```
//in ApplicationFunctionSet_xxx0.cpp
void ApplicationFunctionSet::ApplicationFunctionSet_Tracking(void)
{
.....
 if (function_xxx(getAnalogueexxx_M, TrackingDetection_S, TrackingDetection_E))
   {
    ApplicationFunctionSet_SmartRobotCarMotionControl(Forward, 200);
    timestamp = true;
    BlindDetection = true;
   }
.....
}
```

✚ d: When only the ITR20001 on the right is on the black line and can detect the black line, the car will turn right.

```
//in ApplicationFunctionSet_xxx0.cpp
void ApplicationFunctionSet::ApplicationFunctionSet_Tracking(void)
{
.....
 else if (function_xxx(getAnalogueexxx_R, TrackingDetection_S, TrackingDetection_E))
   {
     ApplicationFunctionSet_SmartRobotCarMotionControl(Right, 200);
     timestamp = true;
     BlindDetection = true;
   }
.....
}
```

✚ e: When only the left ITR20001 on the black line can detect the black line, the car will turn left.

```
//in ApplicationFunctionSet_xxx0.cpp
void ApplicationFunctionSet::ApplicationFunctionSet_Tracking(void)
{
.....
 else if
(function_xxx(getAnalogueexxx_L, TrackingDetection_S, TrackingDetection_E))
   {
     ApplicationFunctionSet_SmartRobotCarMotionControl(Left, 200);
     timestamp = true;
     BlindDetection = true;
   }
.....
}
```

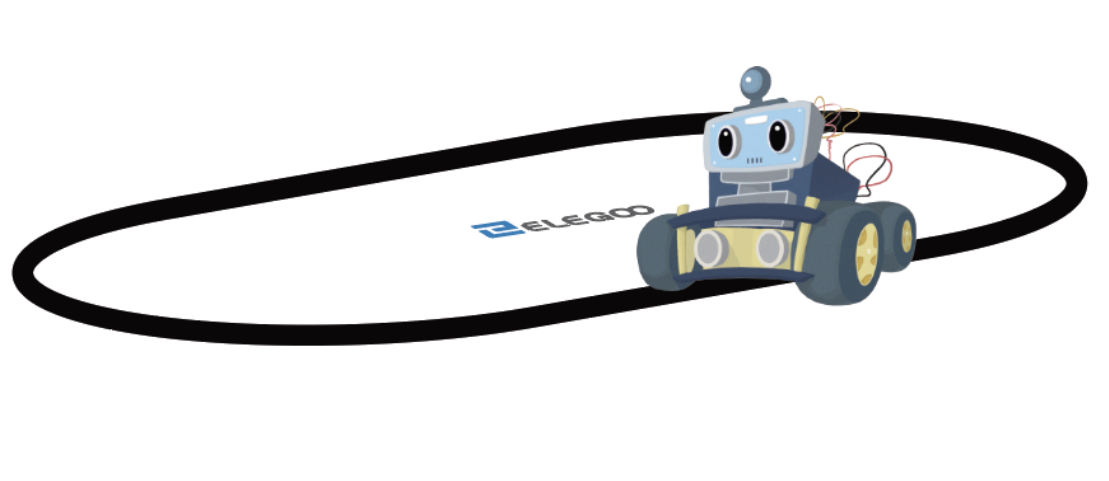
✚ f, g and h are the different conditions under the blind tracking mode.

Let's look at the case of F, when the car is away from the black line, the normal practice is to make the car rotate once to detect the black line, assuming that it rotates once to the left. But in the case of G, with the black lines so close together, is it necessary to rotate it once? Therefore, in blind seeking mode, the car should first rotate from left to right to judge whether there is a black line nearby at a certain angle. If there is no black line detected nearby, it will rotate around to look for the black line.

C:/

```
//in ApplicationFunctionSet_xxx0.cpp
void ApplicationFunctionSet::ApplicationFunctionSet_Tracking(void)
{
.....
 else
   {
    if (timestamp == true)     {
     timestamp = false;
     MotorRL_time = millis();
     ApplicationFunctionSet_SmartRobotCarMotionControl(stop_it, 0);
    }
    /*Blind Detection*/
    if ((function_xxx((millis() - MotorRL_time), 0, 200) ||
    function_xxx((millis() - MotorRL_time), 1600, 2000)) && BlindDetection == true)
    {
     ApplicationFunctionSet_SmartRobotCarMotionControl(Right, 250);
    }
    else if (((function_xxx((millis() - MotorRL_time), 200, 1600))) && BlindDetection == true)
    {
     ApplicationFunctionSet_SmartRobotCarMotionControl(Left, 250);
    }
    else if ((function_xxx((millis() - MotorRL_time), 3000, 3500))) // Blind Detection ...s ?
    {
     BlindDetection = false;
     ApplicationFunctionSet_SmartRobotCarMotionControl(stop_it, 0);
    }
   }
 }
 else if (false == timestamp)
 {
  BlindDetection = true;
  timestamp = true;
  MotorRL_time = 0;
 }
.....
}
```

✚ Upload program. (Please toggle the “Upload-Cam” button to “Upload” when uploading the program.) After uploading, put the car on the black line, the car will move along the black line. Otherwise, it will enter blind search mode to look for the black line, it will turn left and right first, and then revolve to the left.



ELEGOO

http://www.elegoo.com