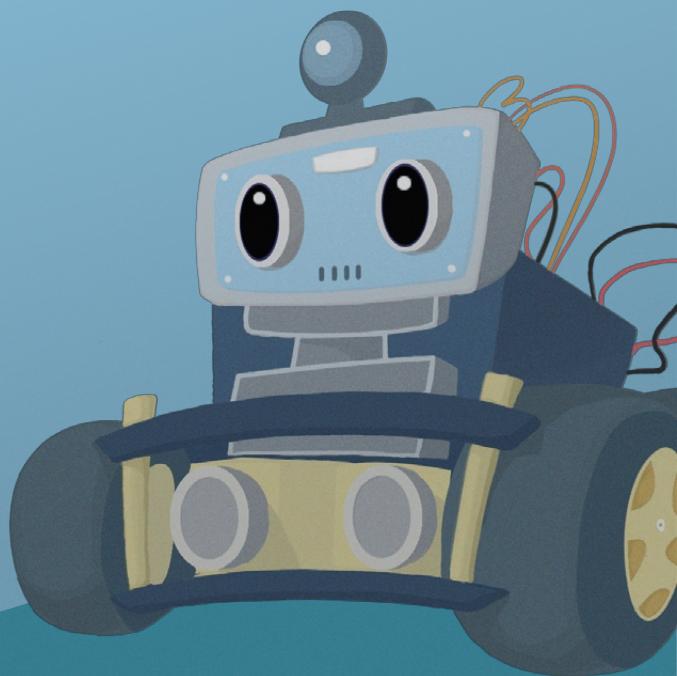


# 3

## SMART ROBOT CAR V4.0 WITH CAMERA



### Tracking Control





## Введение:

- + В этом уроке мы расскажем, как реализовать функцию отслеживания линии в Smart Robot Car V 4.0 и заставить его двигаться в соответствии с нарисованной черной линией.

Если вы хотите создать свой собственный подиум, вот несколько советов.



## Советы:

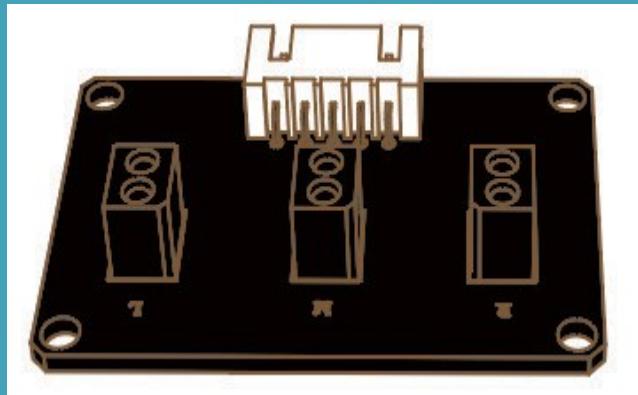
- + Затвердевшая часть линии должна быть как можно более гладкой. Если радиус разворота слишком мал, то автомобиль может не попасть на взлетно-посадочную полосу.
- + В дополнение к отслеживанию линии мы можем расширить наше воображение и разработать другие программы в соответствии с теорией отслеживания линии. Например, можно разработать программу, ограничивающую автомобиль в определенной области независимо от его перемещения.



## Подготовка:

- + Умный робот-автомобиль (с аккумулятором) Кабель USB

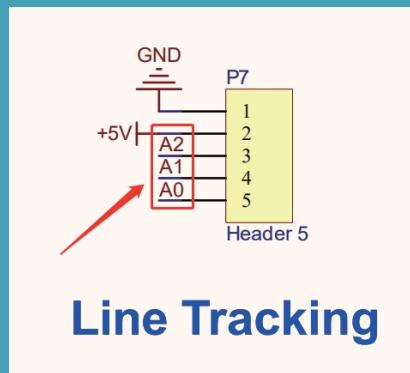
- + В нашем комплекте функцию отслеживания линии выполняет модуль слежения, состоящий из трех фотоэлектрических датчиков и других электрических компонентов.



Для получения подробной информации откройте папку последнего уровня: [Информация о микросхемах -> ITR20001 и SmartRobot-Shield](#)

Фотоэлектрический датчик состоит из инфракрасной парной трубы (передающей/принимающей) и кремниевого фототранзистора NPN, оснащенного источником света и оптическим приемным устройством. Свет, излучаемый источником света, поступает на светочувствительный элемент за счет отражения от измеряемого объекта, после чего путем обработки соответствующих цепей получается необходимая информация. С его помощью можно определять изменение освещенности и тени грунта, его цвет, а также определять наличие или отсутствие близко расположенных объектов.

- + Наконец, посмотрим, к какому выводу подключаются три фотоэлектрических датчика, после чего можно приступать к написанию программы.



 Откройте **Demo1** в текущей папке.

 Прежде всего, определим соответствующие выводы и переменные:

```
C:/ //в DeviceDriverSet_xxx0.h
/*ITR20001 Detection*/
класс DeviceDriverSet_ITR20001
{
    общественность:
        bool DeviceDriverSet_ITR20001_Init(void);
        float DeviceDriverSet_ITR20001_getAnaloguexxx_L(void);
        float DeviceDriverSet_ITR20001_getAnaloguexxx_M(void);
        float DeviceDriverSet_ITR20001_getAnaloguexxx_R(void);
    #if _Test_DeviceDriverSet
        void DeviceDriverSet_ITR20001_Test(void);
    #endif

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

 Перед применением нам необходимо инициализировать вывод и установить три вывода в режим ввода соответственно.

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

- + А затем поместить его в **setup()** для вызова.

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

- + Наконец, данные о работе фотоэлектрического датчика при изменении внешней среды получаются путем считывания аналога с трех выводов.

```
C:/ //в DeviceDriverSet_xxx0.cpp float  
DeviceDriverSet_ITR20001::  
DeviceDriverSet_ITR20001_getAnaloguexxx_L(void)  
{  
    return analogRead(PIN_ITR20001xxxL);  
}  
float DeviceDriverSet_ITR20001::  
DeviceDriverSet_ITR20001_getAnaloguexxx_M(void)  
{  
    return analogRead(PIN_ITR20001xxxM);  
}  
float DeviceDriverSet_ITR20001::  
DeviceDriverSet_ITR20001_getAnaloguexxx_R(void)  
{  
    return analogRead(PIN_ITR20001xxxR);  
}
```

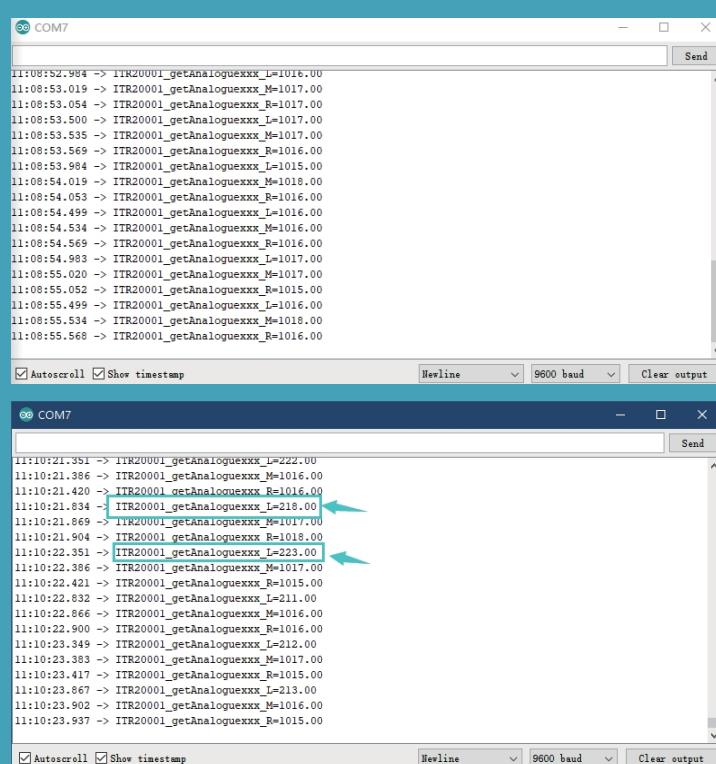
+ А затем вывести его в `loop()`.

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());
}
```

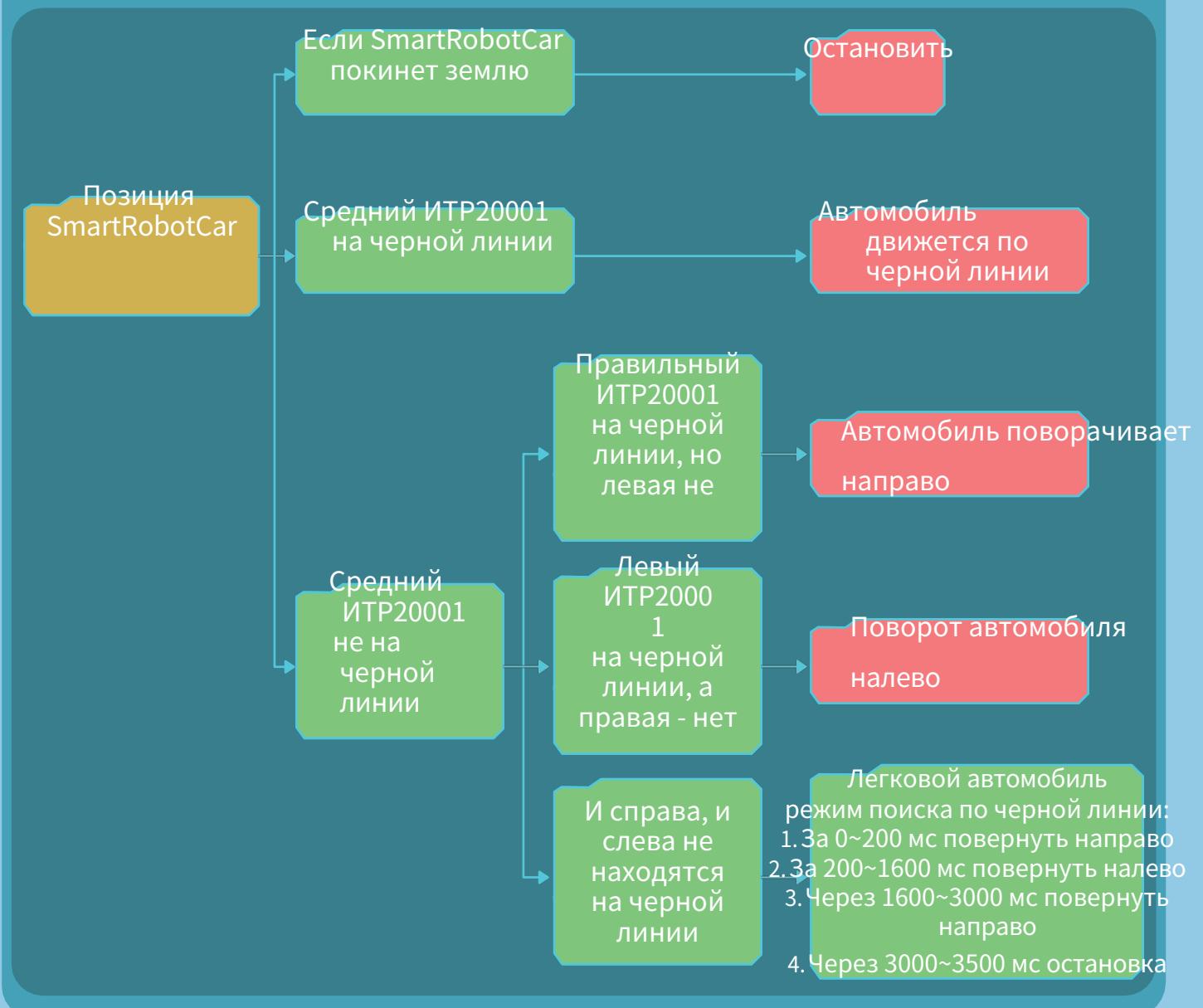
}

Загрузите программу. (При загрузке программы переключите кнопку "Upload-Cam" на "Upload") После успешной загрузки программы откройте последовательный порт и увидите, что данные трех фотоэлектрических датчиков колеблются около 1000. А если заблокировать один из фотоэлектрических датчиков руками, то соответствующее измеренное значение значительно снизится.



⊕ После изучения предыдущих курсов мы освоили использование фотоэлектрического датчика. Далее нам предстоит реализовать функцию слежения, объединив функцию движения автомобиля, изученную в уроке 2, и фотоэлектрический датчик.

⊕ Вначале рассмотрим всю блок-схему режима слежения.



+ Откройте **Demo2** в текущей папке:

+ Прежде всего, рассмотрим относительное определение функции слежения.

C:/

```
//в ApplicationFunctionSet_xxx0.h
класс ApplicationFunctionSet
{
    общественность:
        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;

    общественность:
        boolean Car_LeaveTheGround = true;

    общественность:
        //Если значение, считываемое фотоэлектрическим датчиком,
        //находится в пределах 250 ~ 850,
        //фотоэлектрический датчик находится в
        //черной линии. uint16_t
        TrackingDetection_S = 250; uint16_t
        TrackingDetection_E = 850;

        //Это минимальное значение, полученное фотоэлектрическим
        //датчиком
        //если автомобиль покидает
        //землю uint16_t
        TrackingDetection_V = 950;
};
```

- + Затем необходимо запрограммировать функцию для оценки диапазона.

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;
}
```

- + Затем необходимо запрограммировать ситуацию, когда автомобиль забирается.

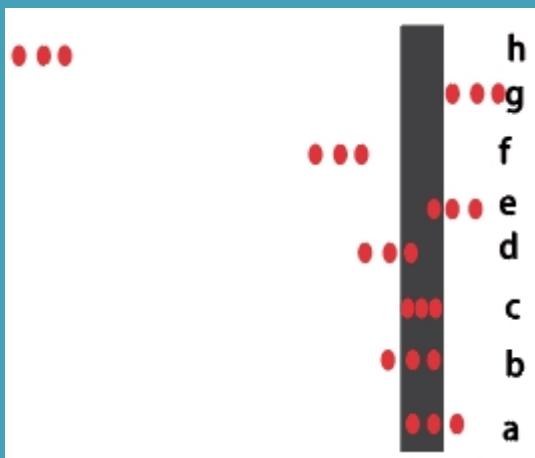
C:/

```
//in ApplicationFunctionSet_xxx0.cpp

static bool ApplicationFunctionSet_SmartRobotCarLeaveTheGround(void)
{
    if (AppITR20001.DeviceDriverSet_ITR20001_getAnaloguexxx_R() > Application_FunctionSet.TrackingDetection_V && AppITR20001.DeviceDriverSet_ITR20001_getAnaloguexxx_M() > Application_FunctionSet.TrackingDetection_V && AppITR20001.DeviceDriverSet_ITR20001_getAnaloguexxx_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;
        }
.....
    }
}
```

+ Подготовительные работы завершены, приступим к написанию функции отслеживания. Проанализируем три сценария работы ITR20001 в режиме онлайн.



+ a, b, c: Когда середина ITR20001 находится на черной линии и черная линия может быть обнаружена, автомобиль будет продолжать двигаться прямо.

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

 d: Когда только ITR20001 справа находится на черной линии и может обнаружить черную линию, автомобиль повернет направо.

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

 e: Когда только левый ITR20001 на черной линии может обнаружить черную линию, автомобиль повернет налево.

```
C:/ //in ApplicationFunctionSet_xxx0.cpp
void ApplicationFunctionSet::ApplicationFunctionSet_Tracking(void)
{
.....
иначе если
(function_xxx(getAnaloguexxx_L, TrackingDetection_S, TrackingDetection_E))
{
    ApplicationFunctionSet_SmartRobotCarMotionControl(Left, 200);
    timestamp = true;
    BlindDetection = true;
}
.....
}
```

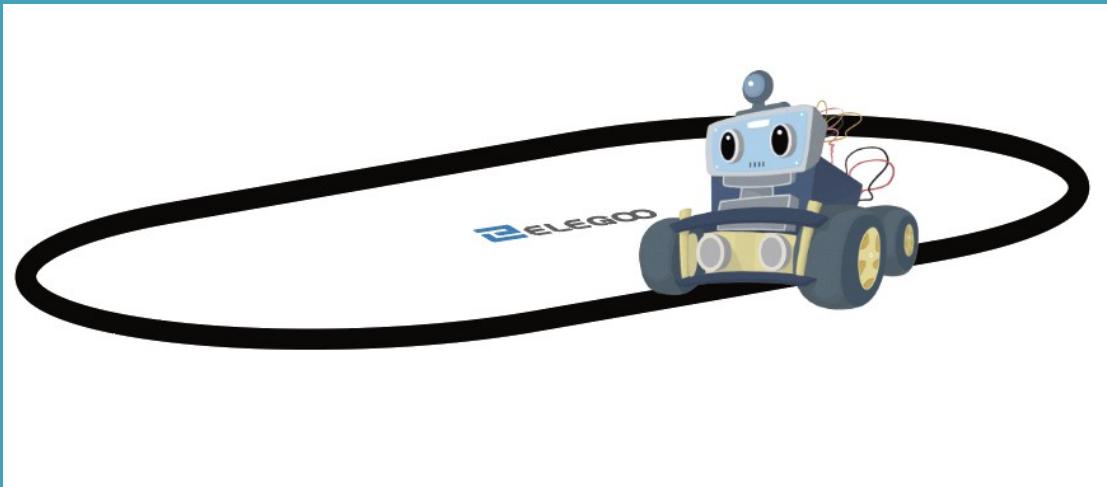
 f, g и h - различные условия в режиме слепого слежения.

 Рассмотрим случай F. Когда автомобиль находится вдали от черной линии, обычная практика заключается в том, чтобы заставить автомобиль повернуться один раз, чтобы обнаружить черную линию, предполагая, что он поворачивается один раз влево. Но в случае G, когда черные линии расположены так близко друг к другу, нужно ли поворачивать машину один раз? Поэтому в режиме слепого поиска автомобиль должен сначала повернуться слева направо, чтобы определить, есть ли поблизости черная линия под определенным углом. Если черная линия поблизости не обнаружена, то автомобиль поворачивается в поисках черной линии.

C:/

```
//in ApplicationFunctionSet_xxx0.cpp
void ApplicationFunctionSet::ApplicationFunctionSet_Tracking(void)
{
.....
else
{
    if(timestamp == true)    {
        timestamp = false;
        MotorRL_time = millis();
        ApplicationFunctionSet_SmartRobotCarMotionControl(stop_it, 0);
    }
    /*Обнаружение слепоты*/
    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))) // Слепое обнаружение ...s ?
    {
        BlindDetection = false;
        ApplicationFunctionSet_SmartRobotCarMotionControl(stop_it, 0);
    }
}
}
else if (false == timestamp)
{
    BlindDetection = true;
    timestamp = true;
    MotorRL_time = 0;
}
.....
}
```

**⊕** Загрузите программу. (При загрузке программы переключите кнопку "Upload-Cam" на "Upload"). После загрузки установите автомобиль на черную линию, автомобиль будет двигаться вдоль черной линии. В противном случае он перейдет в режим слепого поиска черной линии, сначала повернет влево и вправо, а затем повернет влево.



**ELEGOO**  
<http://www.elegoo.com>