



fischertechnik 

ROBOTICS



TXT *Discovery Set*

14 MODELS
MODELS

Welcome to the fischertechnik World of ROBOTICS	4
Some General Information	5
Electricity	5
About this Activity Booklet	5
Robot, Artificial Humans?	6
ROBOTICS, (Almost) Everything Automatic	7
Component Explanations	8
Encoder Motors	8
XS Motor	9
LEDs	9
Lens Tip Lamp	9
Phototransistor	10
Pushbutton	10
Heat Sensor (NTC)	11
Camera Module	11
ROBOTICS TXT Controller	12
A Few Tips	13
First Steps	14
Starter Models	15
Pedestrian light	15
Hand dryer	16
Temperature Control	18
Barrier	19
"Camera Man"	20
Swiveling Camera	22

Mobile Robots - The Next Challenge **23**

Mobile Robot	23
Hindrance Detector	27
Hindrance Detector with Camera	28
Trail Searcher	30
Detection Robot	33
Soccer Robot with Movement Control	35
Soccer Robot	38

Trouble Shooting **43**

Interface Test	43
Cables and Wiring	43
Loose Connection	43
Short Circuits	44
Power Supply	44
Errors in the Program	44
Camera Function	44
Last Sources for Help	45

And what else can I do? **46**

Welcome to the fischertechnik World of ROBOTICS

Hello!

We are happy you have chosen the "ROBOTICS TXT Discovery Set" construction set from fischertechnik. Because with this construction set you can conduct many interesting experiments and solve exciting tasks.



Read this digital booklet and try the experiments and tasks, to learn step-by-step how you can control and program simple as well as complicated machines and robots using the ROBOTICS TXT Controller from fischertechnik.

Learning is a process of building things up from a foundation and it is not possible to start with the most difficult things right away, even though they may be a little bit more interesting than the more simple tasks. This is why we have structured the experiments and tasks in this booklet so that you learn something different with every new task and can then use this as the basis for the next task.

So don't worry, we will start with small things and then work together to progress to the big robots.

We hope you have a lot of fun and success now experimenting with your ROBOTICS TXT Discovery Set.

Your team from

fischertechnik 

Some General Information

Before we really get started with the construction set, you still need to know a few things. Even though the components we will work with are very robust, if you do not handle them properly, they can be damaged under certain circumstances.

Electricity

As you certainly know, a lot of the components in the ROBOTICS TXT Discovery Set use electric power. And you know it is necessary to be particularly careful not to make any mistakes when working with electrical components. That is why you should always read the assembly instructions very carefully when wiring the electrical components.

Never connect the positive and negative poles directly to one another to prevent a short-circuit. This can damage the ROBOTICS TXT Controller or the rechargeable battery.

The subjects of electricity and electronics are just as interesting as robotics (which is what this construction set is about) and there is a construction set from fischertechnik, which deals specifically with these subjects. If you are interested in this, you will also have just as much fun with our "PROFI Electronics" construction set as with the ROBOTICS TXT Discovery Set.



About this Activity Booklet

This PDF activity booklet has a number of functions, which are not present in the printed booklet and which may already be familiar to you from the Internet.

● Links within the Booklet

When something is mentioned somewhere in the text, which is explained in more detail at another point in this booklet (for example, components), the text appears in dark blue and underlined. You can click on the text to move automatically to the page containing the explanation. This is called a "cross reference."

- **Background Infos**

In some cases in this booklet, there are terms or foreign words, which may require explanation. These terms are displayed in green and underlined. If you touch the text with the mouse pointer, a window appears with an explanation.

- **Link Outside of this Booklet**

A few links require an Internet connection (for example, the fischertechnik web site), or an installed ROBO Pro (for connection to the ROBO Pro online help). These terms are displayed in light blue and underlined.

- **Pictures**

A picture is worth a 1000 words. You have certainly heard this sentence before. And because this certainly contains a lot of truth, you can display a picture by touching the words in brown and underlined to see a picture showing what is meant in the text.

- **The ROBO Pro Icon**

This is always located in the vicinity of tasks. This makes sense, because as soon as you click on it, a suitable example program opens with a possible solution.

click-me.rpp

All example programs are listed under **C:\Programs (x86)\ROBOPro\ Example programs\ROBO TX Automation Robots**. [Translators note: Have these directory names been programmed in English? If not, they will have to remain in German.]

Robot, Artificial Humans?

What is your first thought when you hear the word "robot?" Have you ever seen a robot? In a movie or on television? Or perhaps a real one?

There are many different types of robots. Some robots look a bit like a human, while others have only one or more arms. So, what exactly makes a robot a robot?

The dictionary states: "Robots are stationary or mobile machines, which perform set tasks according to a certain program."



ROBOTICS, (Almost) Everything Automatic

Thus, robots are machines controlled by a program. And we call this control of machines (or in our case models) "ROBOTICS."

The "ROBOTICS TXT Discovery Set" provides you with a wonderful start to learn about this subject. This is because the construction set contains everything you need to build and control many different machines.

You can create the programs for control of the models on a PC with the aid of the ROBO Pro 4.0 (or higher) software and then transfer them to the [ROBOTICS TXT Controller](#) using the USB or Bluetooth connection. The controller then controls or operates the model according to the program you have prepared.

Component Explanations

The construction set contains all of the following

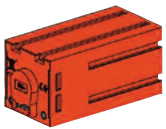
First, it contains numerous fischertechnik building blocks, as well as motors, indicator lights and sensors and colored assembly instructions for building various models.

After you have unpacked all the building blocks, it is necessary to first assemble a few components such as cables and plugs before you can really get started. Details are given in the assembly instructions under "Assembly Tips." It is best to do this first.

Actuators

Actuators are all components, which can perform some type of action. This means that they become "active" in some way when they are connected to electric power. In most cases you can see this directly. A motor runs, an indicator light illuminates and so forth.

Encoder Motors



We use the two encoder motors, contained in the construction set, to drive our robots. At first glance, these are normal electric motors, designed for a voltage of 9 volts and maximum current input of 0.5 amperes.

But the encoder motors can do more: In addition to the connection for the power supply for the motor, they have another connector for a three-pin connection cable, which is used in combination with a so-called encoder to measure the rotation of the motor.

This encoder works the same way as a speedometer on a bicycle. A magnet (in most cases for a bicycle located on one of the spokes) passes by a sensor (attached to the fork of the bicycle in most cases) with each revolution causing the sensor to generate a pulse. These pulses can be counted, and, in the case of a speedometer, for example, multiplied by the circumference of the tire. This gives us the distance traveled.

The encoders on the fischertechnik encoder motors generate three pulses each time the motor shaft revolves once. And because the encoder motors also have a gearbox with a [transmission ratio of 21:1](#)

TXT Discovery Set

(read "21 to 1"), one revolution of the shaft coming out of the gearbox, corresponds to $21 \times 3 = 63$ encoder pulses.

XS Motor



The XS motor is an electric motor, exactly as long and high as a fischertechnik building block. In addition, it is very light. This means, you can install it at points too small for the big motors.

Both gearboxes included in the construction set fit perfectly on the XS motor.

The XS motor is designed for a supply voltage of 9 volts and a maximum current of 0.3 amperes.

LEDs



Two LEDs are contained in the construction set. They can be used in a variety of ways. For example as signal lights in a traffic light, as flashing lights on a robot or for better illumination of an image supplied by the camera also provided in the construction set.

The LEDs are designed for a voltage of 9 volts and consume approximately 0.01 amperes of current.

Caution!

When connecting LEDs to the power supply, always pay particular attention to correct polarity. Connect the positive pole to the red marking on the LED.

Lens Tip Lamp



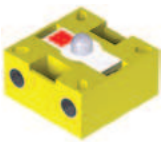
This incandescent bulb contains a lens to focus the light. It looks very similar to an LED. Be careful not to mix them up. On the lens tip lamp the polarity makes no difference - this is why the socket is not marked. You need the lens tip lamp to build a [light barrier](#) in combination with the [phototransistor](#).

The lens tip lamp is designed for a voltage of 9 volts and current of approx. 0.15 amperes.

Sensors

Sensors are so to speak the counterpart to the [actuators](#). This is because they do not perform any actions, but react to certain situations and events. For example, a pushbutton reacts when pressed, allowing an electric current to flow or interrupting its flow. A heat sensor reacts to the temperature in its surroundings.

Phototransistor



Phototransistors are also called "light sensors". This is a "feeler" that reacts to brightness.

For a [light barrier](#) this is the counterpart to the lens tip lamp. When there is a high degree of brightness, that is when the transistor receives light from the lens tip lamp, it conducts electricity. If the beam of light is interrupted, it does not conduct any electricity.

Caution!

When connecting the phototransistor to the power supply, pay particular attention to correct polarity. Connect the positive pole to the red marking on the phototransistor.

Pushbutton



The pushbutton could also be called a touch sensor. Pressing the red button actuates a switch mechanically allowing electricity to flow from contact 1 (middle contact) to contact 3. At the same time the circuit between contacts 1 and 2 is interrupted. So you can use the pushbutton in two different ways:

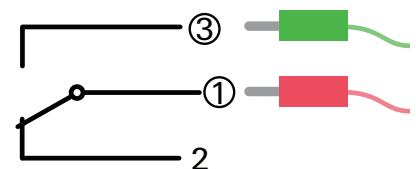
As a "normally open switch" (NO or push-to-make switch)

Contacts 1 and 3 are connected.

[Pushbutton switch pressed](#): Electricity flows.

When the pushbutton is not pressed:

Electricity does not flow



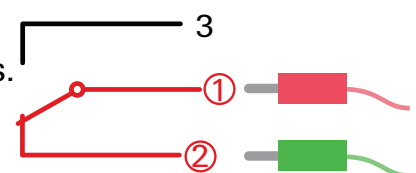
As a "normally closed switch" (NC or push-to-break switch)

Contacts 1 and 2 are connected.

[Pushbutton switch pressed](#): No electricity flows.

When the pushbutton is not pressed:

Electricity flows.



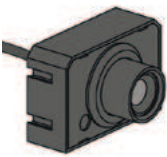
Heat Sensor (NTC)



This component is a heat sensor for measuring temperatures. At 20°C its **electrical resistance** is 1.5 kΩ (kilo ohms, but pronounced 'kil-ohms'). NTC stands for Negative Temperature Coefficient. This simply means that the resistance value decreases when the temperature increases.

The information provided by the sensors, for example, bright/dark, pressed/not pressed and temperature value, can, as we will see later, be transmitted by the [ROBOTICS TXT Controller](#) to a PC where it can be used in combination with the software to program a motor to drive a fan when a **light barrier** is interrupted.

Camera Module



The camera module is a particularly versatile type of sensor. The image resolution is 1 megapixel (meaning that each image consists of one million image dots). Connect the camera to the large USB port (USB1) on your [ROBOTICS TXT Controller](#). The images from the camera can be transferred to the PC and viewed on the monitor. This allows you to see what your robot is doing at any particular time. Moreover the ROBOTICS TXT Controller can process the images thereby recognizing motion, colors and tracks, allowing you to control your robot model accordingly. It is also possible to connect the camera directly to a USB interface on your PC and process the images with the ROBO Pro software. This possibility is also used by a few models.

You can focus the camera image by turning the camera lens.

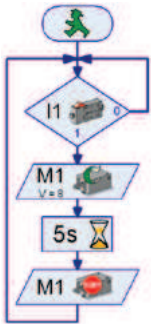
ROBO Pro 4.x Software

ROBO Pro is a graphic programming interface for creating programs for the [ROBOTICS TXT Controller](#).

A "graphic programming interface" allows you to compile programs visually with the aid of graphic symbols instead of "writing" them out by



TXT Discovery Set

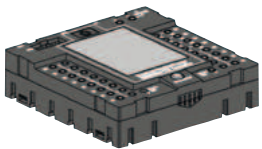


hand line for line. An example of such a program is shown at the left.

The procedure for creating such a program is described in detail in the Chapter "First Steps". The [ROBO Pro Help](#) feature also shows how this works in Chapters 3 and 4.

This software has already been installed on your PC together with this activity booklet.

ROBOTICS TXT Controller



The ROBOTICS TXT Controller is the heart of this ROBOTICS construction set. It controls [actuators](#), and evaluates the information from the [sensors](#).

For this purpose the ROBOTICS TXT Controller has numerous terminals for connection to the components. The instruction manual for the ROBOTICS TXT Controller describes which components can be connected to which connections and the functions of the connections.

The color touch screen allows convenient operation of your ROBOTICS TXT Controller. The camera contained in the construction set can be connected to the USB host port (USB-1). The integrated Bluetooth and WLAN interface is a particularly interesting special feature. It allows you to complete a wireless link between your PC and the ROBOTICS TXT Controller.

You can define how the controller interacts with the individual components and what they are to do in detail in the program you write with the [ROBO Pro software](#).

Power Supply (not included)



As you know, many of the components in the ROBOTICS TXT Discovery Set need electricity to operate, so naturally you also need a power supply.

The fischertechnik Accu Set is best suited for this. It is not included in the construction set.

A Few Tips

Experimenting makes the most fun when the experiments also work. This is why you should follow a few basic rules when building the models.

Work carefully

Take your time and look precisely at the assembly instructions for the model. If you have to look for an error later, this will take much longer.

Check the movement of all parts

When putting models together continually check to see if parts, which have to move, move easily.

Use Interface Test

Before starting to write a program for a model, you should test all parts connected to the [ROBOTICS TXT Controller](#), using the [interface test](#) feature in ROBO Pro. How this works is described in the [ROBO Pro help](#) in Chapter 2.4.

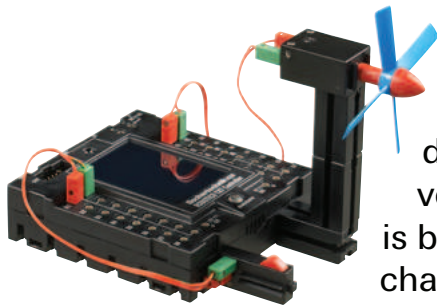
First Steps

Now that you have made all of the preparations and read the information, you can finally start working.

This chapter describes how to:

- build the first simple model, a ventilating fan, and connect it to the [ROBOTICS TXT Controller](#),
- connect the ROBOTICS TXT Controller to the power supply and PC,
- load the [ROBO Pro Software](#) and test the model,
- load and start a ROBO Pro program, and
- create and start your first simple program with ROBO Pro

[Click here](#), to see the first, easy-to-understand steps.



Since you will be working particularly with the [ROBO Pro Software](#) in addition to the fischertechnik components themselves, you should be familiar with the details for writing programs. And because this is explained very clearly in Chapters 3 and 4 of the [ROBO Pro Help](#), it is best at this point to continue by working through these chapters carefully.

The following tip also applies here: Take your time and concentrate; then you will have that much more fun with the models later.

Starter Models

After reading through Chapters 3 and 4 in the [ROBO Pro Help](#), you will be able to program some of the models in the construction set. Let's get started. ROBO Pro has various levels, which you can select on the menu bar. We will start with very simple programs at Level 1. Whenever you have finished building and wiring a model, check whether all inputs and outputs on the [ROBOTICS TXT Controller](#) are properly connected and if the [sensors](#), [motors](#) and [lights](#) all function properly with the aid of the [interface test](#).

Pedestrian light

A pedestrian light has been installed in front of your house. Since the technician from the traffic light company doesn't have much time, you offer to program the traffic light control for him.



The man explains to you how the control is supposed to work. But first, build the model.

Task: (Level1)

The traffic light should be red initially. When pushbutton I1 is pressed by a pedestrian the traffic light should change to yellow three seconds later and after an additional four seconds to red. The green phase is to last for 10 seconds, before the light turns red again.



Programming Tips:

The various LEDs are associated with the following outputs on the TXT Controller.

- Red – M1
- Green – M2

Turn the indicator lights on and off one after another to obtain the desired sequence.

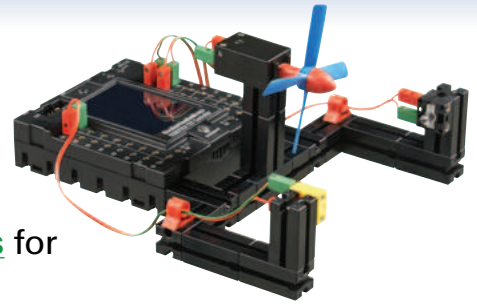
You can load the finished program by clicking on the picture on the right.



Pedestrian_light.rpp

Hand dryer

In the bathroom at your school, new hand dryers have been installed next to the sinks. These are equipped with [light barriers](#) for switching the fan on and off.



First, build the model as described in the assembly instructions.

Task 1: (Level 1)

Now it is necessary to program the hand dryer so that as soon as the light barrier is interrupted the fan switches on and then back off again after 5 seconds.



Programming Tips:

In the program sequence first switch on the light for the [light barrier](#) at [output M2](#).

Then wait one second to allow the phototransistor time to react to the light. The light barrier should then work properly.

Then check (interrogate) the [phototransistor](#) at [Input I1](#). If the value is "1" (light barrier not interrupted), the input should be interrogated continuously in a loop.

As soon as the value changes to "0" (light barrier interrupted), switch motor M1 on and then back off after 5 seconds.

Then return to the loop for interrogating the phototransistor.

Start the program with the [start button](#) and check whether it works the way you want. If it does work right, you're on your way to becoming a professional ROBO Pro programmer.

If it doesn't work, try to find out why.

The [interface test](#) allows you to check whether all inputs and outputs work properly and if they are connected correctly.

While the program is running, you can follow the program sequence with the red building blocks. This allows you to quickly see where an error has crept in.

Finally you can compare your program with the finished example program, which can be called with the icon on the right.



Hand dryer_1.rpp

After you have taken this hurdle, we want to change the task slightly:

Task 2: (Level 1)

The school principal is always interested in saving energy and doesn't like the hand dryer to continue running after your hands are dry. He asks you to rewrite the program so that the fan shuts off as soon as you pull back your hands. That's not a problem for you, is it?



Programming Tips:

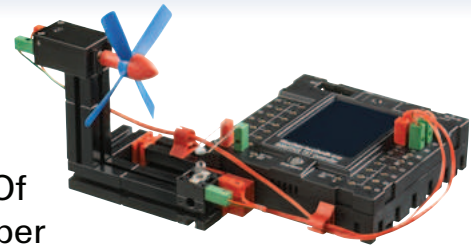
As in the first program, it is necessary to interrogate the [phototransistor I1](#). If the value is "0," switch motor M1 on and if the value is "1" switch motor M1 off and so forth.

For this task, there is also a complete program just in case:



Hand dryer_2.rpp

Temperature Control



At home where you live, a new air conditioning system has been installed. Of course, you immediately asked the plumber how the temperature control works. He was happy to explain to you that a temperature sensor continually measures the temperature. As soon as a maximum value is exceeded, the air conditioning is switched on. However, if the temperature is below the minimum value, the air conditioning is switched off and the heating turned on. Now you need to try to program a [control circuit](#) based on the "temperature control" model. But first, build the model.

Task: (Level 1)

The heating is simulated by lens tip lamp M2. The fan at [output M1](#) serves as the "air conditioning unit". An NTC thermistor at [input I8](#) is used to measure the temperature.

Program the model so that above a certain temperature value the heating is shut off and the fan is turned on. This is to cool the house until a minimum value is reached. Then the fan is to be shut off and the heating turned on.



Programming Tips:

Please note! The resistance of the [NTC thermistor](#) decreases with increasing temperature. Therefore, the upper temperature limit is the smallest value for I8. The fan should switch on when this limit is reached. The lower temperature limit is the largest value for I8. The heating should switch on when this limit is reached.

You can see the value of I8 at room temperature using the [interface test](#). Turn on the indicator light M2 and observe how much the value decreases. Now switch on the fan to find out how much the value increases. Use this as a basis for selecting the limits for heating and cooling.



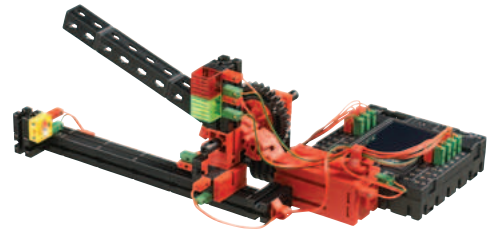
You can open the finished program by clicking on the icon on the right.

Temperature_control.rpp

Barrier

Such a barrier is frequently located at the entrance or exit to a parking lot. First, build the model as described in the assembly instructions.

Our barrier could be at the exit.



Task: (Level 2)

When a car approaches the barrier, it interrupts a light barrier. This should open the barrier, leave it open a certain time (e.g. 5 seconds) and then close it again; however only when the light barrier is not interrupted, because otherwise the barrier could damage a car located exactly below the bar. The traffic light should turn green to show the car driver that the barrier is open and red when it is closed.



Programming Tips:

Use a subroutine for the "opening" and "closing" operations for the barrier (see also [ROBO Pro Help](#) Chap. 4).

For this purpose set ROBO Pro to Level 2.

You can open the finished program by clicking on the icon on the right.

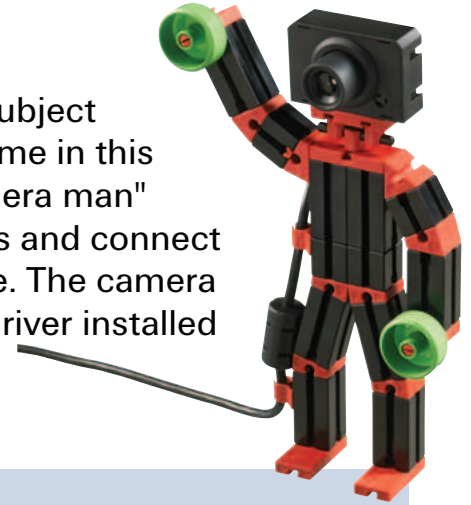
Barrier.rpp



"Camera Man"

Now we want to take a look at the interesting subject of cameras and image processing for the first time in this activity booklet. For this purpose build the "camera man" model as described in the assembly instructions and connect the camera directly to your PC with a USB cable. The camera will be recognized and the associated camera driver installed automatically.

Now start the ROBO Pro software.



Task 1:

Place the camera man in front of your PC and let ROBO Pro show you what the camera "sees".



Programming Tips:

To do this it is not even necessary to write a ROBO Pro program. Simply open a new program in ROBO Pro, go to the "camera" tab, set the camera connection to "PC" there and click "Switch on camera".

You can focus the image by turning the lens on the camera.

Task 2: (Level 3)

Now you can program the camera man to react to motion. As soon as someone walks through the image, a red light connected to output M1 should flash three times on the TXT Controller.

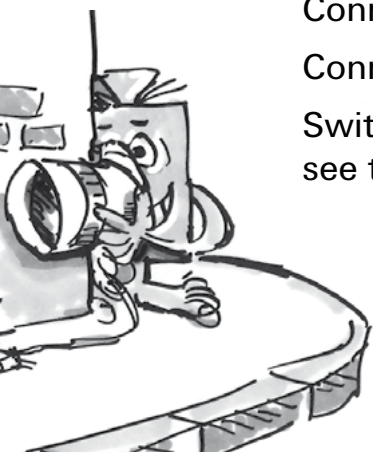


Programming Tips:

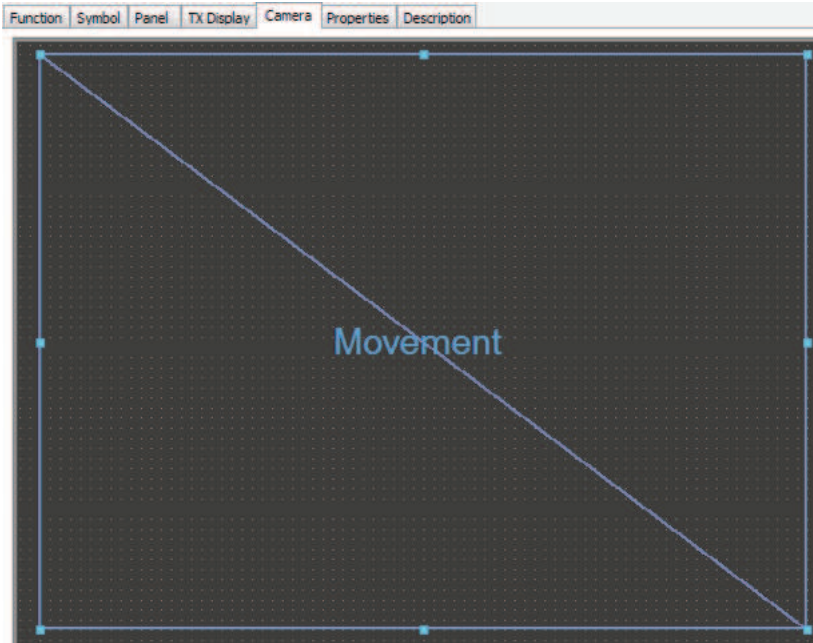
Connect the TXT Controller to your PC using a second USB interface.

Connect an LED with red cover to output M1 on the TXT Controller.

Switch to level 3 in ROBO Pro. Refer to the [ROBO Pro Help](#) in Chap. 5 to see the additional functions available there.




Draw a rectangle of the desired size in the camera "Motion" sensor field in ROBO Pro.



The interactive field for motion recognition, which can be opened by right-clicking with the mouse on the inserted sensor field, allows you to set how intense the motion must be to actuate the sensor and how extensive the motion must be in relation to the entire sensor field, for it to react. The best way is to try it with the default values first and then later change the settings to see the reaction.



Create a routine in ROBO Pro for checking whether the sensor field has recognized a motion. To interrogate the sensor field, use the camera input element from Level 3 in ROBO Pro and use a loop to check whether motion is present.

Interrogation of "Motion C" input  (Change contrast). Is the value >0? If so, the light should flash.

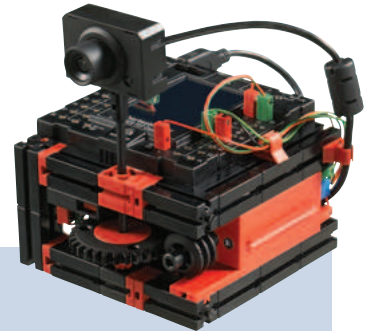
Further information on the camera sensor fields is given in [ROBO Pro Help](#) Chap. 11.

You can open the finished example program [here](#).

Camera_man.rpp

Swiveling Camera

Next you can build a surveillance camera for your room. First, build the model as described in the assembly instructions.



Task: (Level 3)

This camera is driven by an [encoder motor](#) to repeatedly turn it slightly before checking if anything moves in the room. If so, it should trigger an alarm on the TXT Controller loudspeaker; if nothing moves, it should continue to turn slightly before checking again.

After reaching the end of the swivel range, it should return to its initial position in small increments.



Programming Tips:

The camera should first move to its initial position (until pushbutton I1 is depressed).

Then it should repeatedly move a certain number of pulses in the other direction. You can easily check how many pulses are practical per rotation with the aid of the [interface tests](#).

Use the camera "Motion" sensor element to check whether anything changes in the image. If so, trigger an alarm on the TXT Controller loudspeaker, see ([ROBO Pro Help](#)).

Repeat this operation until the camera reaches the end of its swivel range, then move back in the other direction to limit switch I1.

Here is our solution for this task:

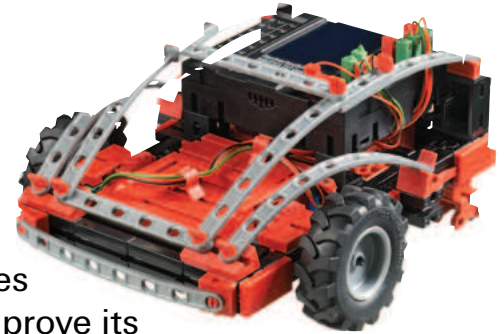
Swiveling_camera.rpp

This model can be operated independently of the PC in the download mode.

It is the perfect surveillance camera to keep trespassers out of your room.

Mobile Robots - The Next Challenge

Mobile Robot



With this model, we want to find out together how you can control a moving robot. How do you get it to move, how does the steering work and can you perhaps improve its precision? We will answer these questions with the help of the tasks in this chapter.

But first of course, you have to put the robot together. As always, you will find the description in the assembly instructions.

Take your time putting it together. Look closely at the drawings in the assembly instructions and the wiring as well. If you do not connect the components to the [ROBOTICS TXT Controller](#) as described in the assembly instructions, the robot may not behave like you expect.

After assembly, check all components connected to the ROBOTICS TXT Controller with the ROBO Pro software [interface test](#). When the motors turn counterclockwise, the robot should move forwards.

Direction of motion	Direction of rotation, motor 1	Direction of rotation, motor 2
Forwards	Left	Left
Backwards	Right	Right
Left	Left	Right
Right	Right	Left
Stop	Stop	Stop

Task 1: Simply Straight Ahead

Have the robot move straight ahead for 3 seconds (not on a table, danger of falling off!) and then straight back for 3 seconds.

Did the robot really come back to its starting point?

Repeat the program several times and observe if the robot really moves precisely straight ahead and back.



Programming Tips:

Even if this task is easy for you, we would like to make a few suggestions:

Mobile_robot_1.rpp

The Steering

Even if it is fun to watch the robot move straight ahead, it is somewhat monotonous. That's why, it is now time to learn to move in a curve. And how is that done? Very simple:

Task 2: Also moving in a curve

Let the robot move straight ahead again for three seconds (both motors turning at the same speed) then change the direction of rotation of the right motor (M1) for one second and then let the robot move straight ahead again for three seconds (meaning both motors at the same speed in the same direction).

Find out how long you have to let the motors run in different directions to make the robot turn 90°.



Programming Tips:

For this, change the waiting time after the command for changing the direction of the second motor.

You can open the finished program as usual by clicking on the icon on the right.

Mobile_robot_2.rpp

Task 3: Moving through a figure

Now that you know how long it is necessary to reverse the direction of rotation of a motor to make the robot turn to the right or left, program the robot so that it moves in a rectangle, returning to its starting position.

Make a mark to check if the robot really moves precisely to its starting position.



Programming Tips:

You can create a subroutine for "turning a corner." This keeps the main program clearer.

You probably already have the solution to the task in your head. But here is another recommendation, just in case:

Mobile_robot_3.rpp

Always the same, but not exactly?

You have probably noticed that the [repetition accuracy](#) of the robot could be improved. Even when it performs precisely the same task several times, the result is not always the same. This is due to various reasons. One of these is that both motors do not rotate at exactly the same speed. For example, the gearbox for one motor can run with more friction than the other motor. And because both motors are operated at the same voltage (nine volts), this makes one motor rotate slower than the other. Since, in the past, we have controlled our robot using waiting times, perhaps one wheel has rotated farther during this time than the other one.

Thus, the solution would be to have both motors rotate at exactly the same speed. And this is precisely what encoder motors do.

Task 4: Using Encoder Motors

Repeat the last task using the [encoder motor elements](#) instead of the normal [motor output](#) and [waiting time elements](#). How to use these is described in the [ROBO Pro help](#) in Chap. 12.6.



Programming Tip:

With the encoder motor element you can control both motors simultaneously with one program element. With the [distance entry field](#) you can set each motor to turn exactly the distance it is supposed to.

You can open the solution suggestions as usual by clicking on the icons on the right.

Mobile_robot_1_sync.rpp

Mobile_robot_2_sync.rpp

Mobile_robot_3_sync.rpp

An additional program element is not required in ROBO Pro to count the pulses at the high speed [counter inputs C1-C4](#). The counter input C1 is automatically assigned internally to motor M1; M2 is connected to C2 and so forth.

Note: If the model does not move straight ahead in spite of using the [encoder motor elements](#) the cause can be the model itself. For example, if a hub nut, which transfers the force from the axle to the wheels, is not tight enough the axle slips and the model moves in a curve even though the motors are rotating at the same speed. Therefore be sure the hub nuts are tightened well.

Hindrance Detector

Your robot can now move straight ahead and turn. And up to now it does this precisely as described in your program.

However, a robot should actually be able to react as independently as possible. This is why we now want to have it react to obstacles with the aid a bumper.



First, build the model as described in the assembly instructions.

Task:

The hindrance detector should move straight ahead. As soon as it hits an obstacle with its left bumper, it should stop, move back a short distance, turn slightly to the right and then continue to move straight ahead. If it hits an obstacle with its right bumper, it should move around the obstacle to the left in the same manner.



Programming Tips:

Left bumper: Pushbutton I6; Right bumper: Pushbutton I5

Use a subroutine for each of the various actions: forwards / reverse, go around to left / right.

Ensure that the robot does not move the same number of pulses when moving to the right and to the left; otherwise it could get into a corner and not be able to get out. However if the number of pulses is different, it will be able to work its way out of a corner.

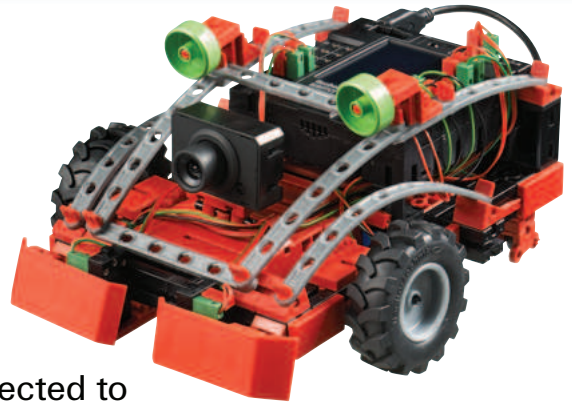
You can find the finished program here:

[Hindrance_detector.rpp](#)



Hindrance Detector with Camera

Now you can equip your mobile robot with a camera so it can see where it is going and allow you to control it by remote control. To do this simply take the hindrance detector and install a camera as described in the assembly instructions. The camera can be connected to the [USB-1](#) interface on the TXT Controller.



Task 1:

First the robot should behave in exactly the same manner as the hindrance detector without camera. In addition it should stop and turn around when you hold a red card in front of it.



Use the camera "Color" sensor field for color recognition and draw it large enough that its area covers the major portion of the camera image.

Then set the robot parameters so that it moves straight ahead as long as a red card is not detected and no hindrance is present. You can use the hindrance detection from the "Hindrance detector program".

It is best to use your own subroutine for checking the color. Check the camera input to determine if the red value is greater than green and blue and if the color is lighter than black.

Use the small red card cut from the large cutting template in the construction set.

The "sensor values" in the camera window in ROBO Pro show the color value for the red card.

Again you can see our solution here:

Note:

Hindrance_detector_camera_1.rpp

Avoid direct sunlight. The robot may confuse sunlight with the red card. Particularly bright sunlight with a high percentage of red.

Task 2: Remote control of robot

Note: This task requires a WLAN connection between the TXT Controller and your PC. The process for completing this connection is described in the instruction manual for the TXT Controller.



Tips:

After successfully completing the WLAN connection, open the appropriate program with the icon.

Change to the control panel and look for the interface for remote control of the model there.

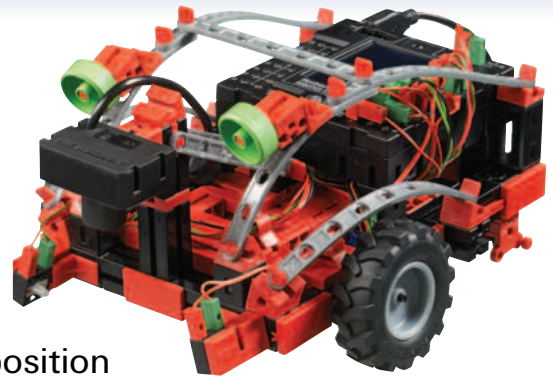
Hindrance_detector_camera_2.rpp

Start the program in the online mode. Now you can use the various buttons on the remote control to control the robot and you can also see where it is going. If you overlook an obstacle, it will be detected by the bumper and the model will automatically move back a short distance.

Have fun exploring the area!

Trail Searcher

With this model the camera is used to make the robot move along a line. The amazing part of trail searching with the camera is that, in addition to recognizing whether a track is present, it can also detect its precise position and output it in the camera image. This allows the robot to react and either move straight ahead, when the trail is exactly in the middle of the image or correct its motion to the left or right, when the trail is not in the middle.



The objective is for the robot to find the black line and move along it. But one thing at a time. First, it is necessary build the trail searcher model. Naturally the assembly instructions describe how to do this.

After you have completed the model, you should check it with the [interface test](#) to ensure that all components are properly connected to the [ROBOTICS TXT Controller](#) and are working correctly.

Task 1: Recognition of a trail

Program the robot so that it follows a straight black trail when placed on it. If it loses the trail or it ends, the robot should stop. Use the obstacle course from the construction set for this task.



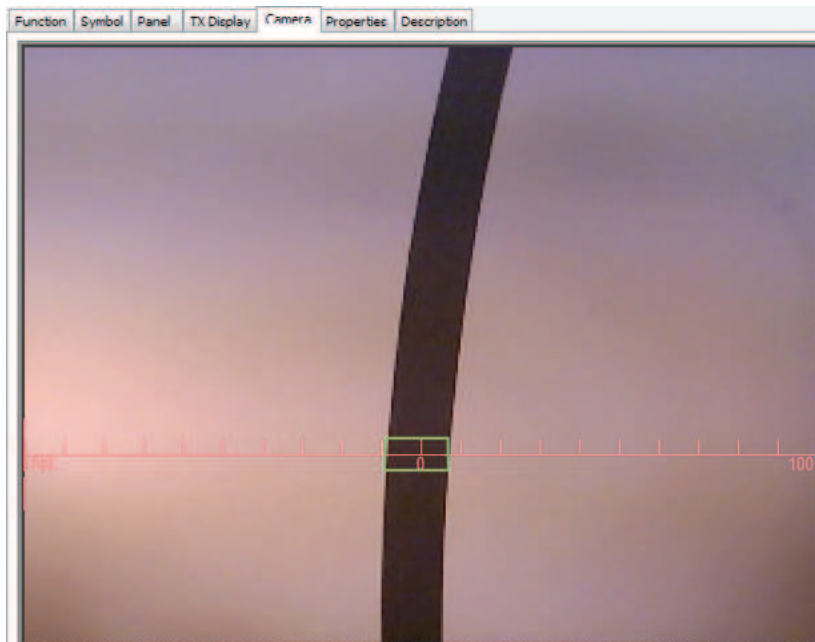
Programming Tips:

Use the obstacle course from the construction set to test the black trail.

Set the camera so that the trail is in proper focus in the ROBO Pro camera window.

Use the camera "Line" sensor field for detecting the trail. It consists of a straight line in the camera image, located across the image from left to right with a scale from -100 to +100. 0 is precisely in the middle.





Further information on the camera "Line" sensor field is given in [ROBO Pro Help](#) Chap. 11.

Now you can check the position of the trail in your program.

- If it is located between -10 and $+10$ the robot should move straight ahead (M1 and M2 = left: $v = 5$)
- Between -11 and -40 it should correct slightly to the left (M1left: $v = 5$, M2 left: $v = 2$)
- Between $+11$ and $+40$ it should correct slightly to the right (M1left: $v = 2$, M2 left: $v = 5$)
- At values < -40 it should correct strongly to the left (M1left: $v = 5$, M2 right, $v = 3$)
- At values $> +40$ it should correct strongly to the right (M1right: $v = 3$, M2 left, $v = 5$)
- If it loses the trail, it should stop.

You can find the finished program here:

Trail_searcher_1.rpp

Your robot can now react. However it would be even better if your robot would turn around at the end of the trail instead of just stopping.

Task 2: Turning around at the end of the trail and following it back

Expand your program to include a function, so that when the robot gets off the trail it turns around and returns.



Programming Tips:

There are several possibilities for correcting the direction. You can stop one motor and let the other one continue to run or have one motor rotate opposite the direction of movement. Experiment to determine which method is better.

Here is our suggested solution:

Trail_searcher_2.rpp

So! Now your robot can move on visual "rails". The only disadvantage is that you first have to set it on the rail. We want to change this. The robot should search for its trail on its own.

Task 3: Finding the trail and following it

Write a "search" subroutine to make the robot search for a black trail if it does not find one when the program starts. For this purpose, the robot is to first move in a circle for one time. If it does not find a trail when it does this it should move straight ahead for a short distance. As soon as the robot detects a trail, it should follow it. If this doesn't, the search should start over from the beginning. After it has moved in a circle 10 times without finding a trail, it should stop and flash three times.



Programming Tips:

In case you are not exactly on the right trail, here is our suggested solution:

Trail_searcher_3.rpp



Detection Robot

On this model we now want to combine a number of camera functions. The robot should be able to look forward to detect its surroundings, however it should also be capable of looking down and following the various colored trails on the obstacle course included. Moreover it should also collect the "temperature data" for its surroundings using its NTC thermistor. First, build the model as described in the assembly instructions.

Task 1: The robot should function as follows:

First place it on the black trail. It should move along it to its end. Then it should turn around and move in the opposite direction. After reaching the end of the black trail where the trail is wider, it should raise the camera and look forward. If you then hold a red or green card in front of it, it should first follow the black trail and then turn on to the red or green trail. Before moving off, it again measures the ambient temperature and sends it to the TXT Controller display.



You can cut the small colored cards out of the large cutting template in the construction set

Programming Tips:

This program is intended to show you everything that is possible with ROBO Pro. Since we know that programming is not all that simple, just load the prepared program and try it out:

Detection_robot.rpp

Task 2:

The "Trail" subroutine is responsible for controlling the encoder motor while following a trail. Compare it with the subroutine from the trail searcher model. What differences do you see?



Solution:

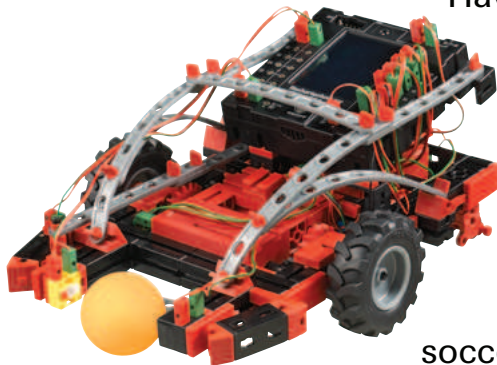
The subroutine for the detection robot is considerably more complicated than for the trail searcher. It contains a control function which makes the degree to which the direction is corrected dependent on the distance from the middle of the trail. In comparison, the trail searcher differentiated only between straight ahead, minor correction and major correction. Creating such a control function requires a great deal of programming experience (such as our professional programmers have). But the control results are also significantly better than with the trail searcher model. In the future you can simply use this subroutine for your own programs, and be happy that it works so well.

Task 3:

If you have a smartphone (with Android operating system), you can use it to control the detection robot. You can download the app for this from the Google Playstore. It is called "TXTCamdroid". Connect the TXT Controller to your smart phone via WLAN to get started. You can see what the camera sees on your smartphone display. You can control the model with different keys and send it on discovery trips. Have fun!



Soccer Robot with Movement Control



Have you ever heard about the Robo Cup? This is the Soccer World Cup for robots. It is held in a different country every year. There are various leagues for various types of robots. Detailed information is available on the Robo Cup homepage <http://www.robocup.org>

Two suggestions for construction of a soccer robot are given in the assembly instructions.

It is just as agile as our other robots, but in addition it has a light barrier for detecting a ball and triggering a "kicking mechanism." Now we can equip and control the robots with the camera in a number of different ways.

First, build the soccer robot with movement control as described in the assembly instructions.

Then you can program it and "train" it to perform a few ball tricks. And, as always you should use the [interface test](#) to see if the model functions properly before starting the programming.

Use the orange table tennis ball included in the construction set for the ball.

Note: It may be necessary to adjust the pushbutton, used as a limit switch for the lever on the actuating mechanism, so that it is depressed when the lever is forward, however does not block the lever simultaneously.

With this robot the camera is located on its own stand next to the model, not installed on the model itself. It should be plugged directly in to the USM port on the PC.

Task 1: "There, he has the ball and shoots . . ."

The first step is to teach the electronic ball wizard to shoot the ball as soon as it is detected by the [light barrier](#). Experiment a little with the "shooting speed." A short pause between "detection" and "shooting" may possibly lead to an improvement.

**Programming Tips:**

For this task it is first necessary to connect the TXT Control to your PC using the USB cable.

As with the [hand dryer](#) you should wait a few seconds after the [lens tip lamp](#) switches on before checking the [phototransistor](#) on the light barrier.

Being a trainer is not easy. If your robot player does not obey you, perhaps you can get its attention with our program suggestion.

Soccer_robot_movement_control_1.rpp

But since a real ball wizard should be able to do more than just a penalty shootout, we want to expand the capabilities of our soccer robot a bit.

Task 2: Motion control for soccer robot

Now we want to program the soccer robot to react to motions you make in front of the camera. When you wave with your left hand, it should move to the left; if you wave with your right hand it should turn to the right. If you wave with both hands in the middle of the image, it should move straight ahead. When the light barrier detects the ball, it should shoot, naturally in to the goal if possible.

**Programming Tips:**

Note: For this task it is necessary to connect the TXT Control to your PC via WLAN or Bluetooth.

Even though this task may sound somewhat difficult initially, you have already used most of the functions for this program with your other robots.

Use three camera "Motion" sensor fields for control of the motion. Position one of these in the middle of the camera image, one on the left and one on the right. Place the camera in front of your monitor where you can easily reach the three areas in the camera window with your arms without the sensor fields reacting unintentionally. For this purpose place the fields preferably in the upper half of the camera surface.

Then simply have your program interrogate the three sensor fields to determine which of the fields has detected motion and actuate the robot accordingly to move to the left, right or straight ahead. You have already programmed the shooting mechanism in Task 1. You can integrate this into your program.

Then start the program in the ROBO Pro online mode. The camera image is transferred to the PC over the USB interface; the robot control is wireless via Bluetooth or WLAN.

For this model you only need the goal from the soccer stadium included with the construction set. You could also use the "side boards" to keep the ball from rolling away. However they are actually required for orientation of the robot only after building the next model. And now, have fun "waving" and shooting goals.

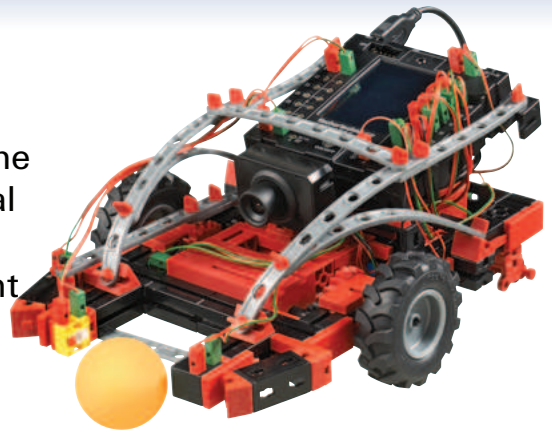
Once again we also have a recommended solution for this program.

Soccer_robot_movement_control_2.rpp

* see also [ROBO Pro Help](#), Chap. 11.

Soccer Robot

Although it is very amusing to control the soccer robot by waving your arms, a real soccer robot should be able to kick the ball in to the goal by itself. Now we want to teach him to do this by installing the camera directly on the robot.



Once again the assembly instructions describe how to do this.

Connect the camera to the [USB-1](#) interface on the TXT Controller.

The entire soccer stadium is required for this model. It consists of the unprinted, white rear, you already used for the obstacle course for the trail searcher and the detection robot as well as the sideboards printed with black stripes and the goal.

The robot needs the pure white background to detect the orange ball with the camera "Ball" sensor field.

The black stripes on the sideboards, which have a different appearance on each side, allow the robot to detect where it is on the playing field and where the goal is. The stripes on the rear wall of the goal show the robot where to shoot, when it has the ball.

Task 1:

As with the detection robot, this robot is not easy to program. Therefore start by trying out the program provided by loading it on the TXT Controller.



Here is our suggested solution:

Soccer_robot.rpp

Tips:

Adjust the lens on the camera so that the stripes on the goal sideboards in the stadium are focused when the robot is at the opposite end of the stadium.

Position the robot in the stadium and then throw a ball on to the playing field. The robot will search for the ball, attempt to catch it and shoot it in the direction of the goal as soon as it has turned to face the goal. With a little luck the robot will shoot the ball in to the goal.

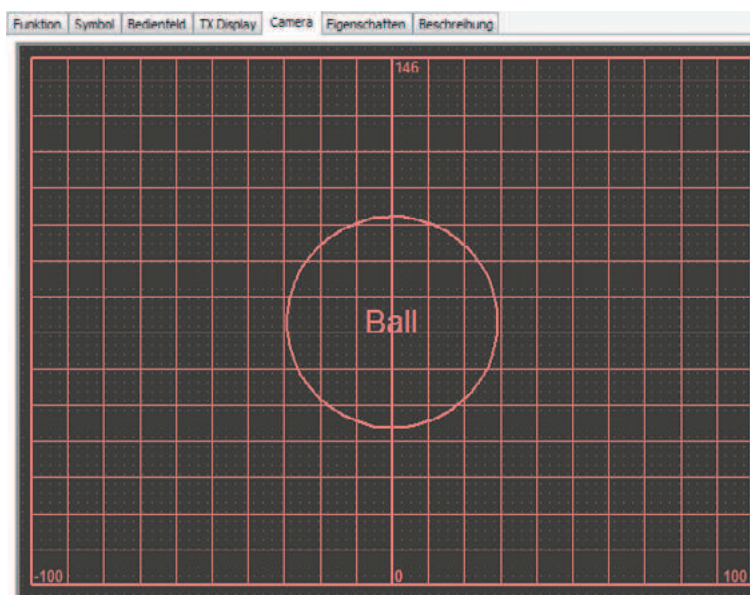
Task 2:

Try to understand how the ball detection works based on the description in the ROBO Pro help.



Solution:

The "Ball" sensor field detects a colored ball on the white background. It outputs the position of the center point of the ball in the sensor field. See also [ROBO Pro Help](#), Chap. 11.



Similar to the trail searcher, 0 is exactly in the middle between right and left. In the vertical direction the value goes from 0 to an automatically calculated maximum height. The robot selects its direction of motion depending on where the ball is located.

Task 3:

Which camera sensor element does the program use to evaluate stripes on the sideboards?



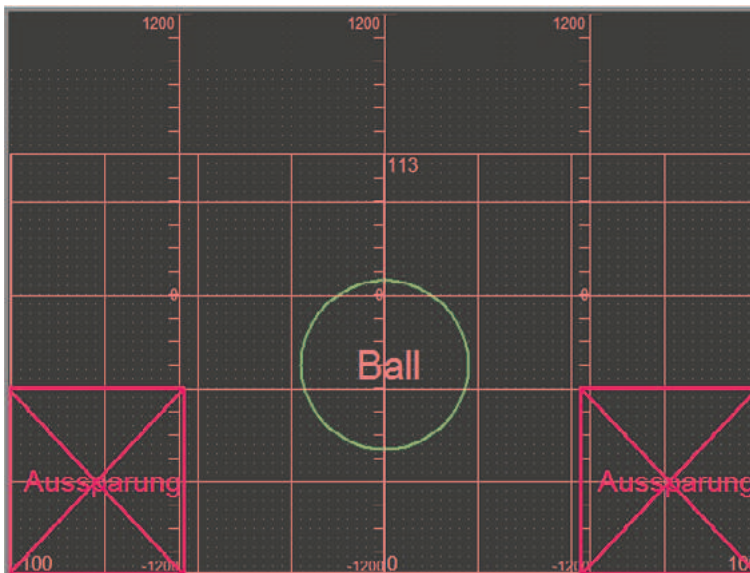
Solution:

The "line" sensor element is used. In contrast to the trail searcher, three line sensors are located from top to bottom in the camera window to detect the three horizontal stripes on the sideboards. The robot uses their width to calculate where it is presently located and where it needs to move to.

Moreover the stripes allow it to recognize when it is too close to the sideboard and to stop before colliding with the sideboard.

Task 4:

Two "Exclusion" sensor fields are present in the ROBO Pro camera window for this robot. Can you imagine what these are for?



Solution: These sensor fields supplement the sensor field for ball recognition. The exclusion areas are not evaluated when looking for the ball. This can be helpful, when objects such as the [phototransistor](#) for the soccer robot are located in the image and could be mistaken for the ball.

How does the robot find its way around the playing field?

Maybe you have wondered how the robot finds its way around the playing field using the stripe codes on the sideboards. As a matter of fact, this is not a simple operation, but it works according to the following principle.

In fact the robot is capable of determining its position on the playing field to an accuracy of approximately 2 cm and an angle of approx. 5 degrees based on the sideboards. For this purpose the robot measures the height of the sideboards at various points and uses this height to calculate the distances and then finally the angle and absolute position. This process is known as triangulation.

The sideboards always have 3 black stripes with 2 white stripes in between. One of the black stripes is wider than the other stripes. This allows the robot to recognize which sideboard is which. All totaled there are 5 different sideboard patterns, one for each side of the playing field and one for the goal.

Moreover the robot uses the counting pulses from the encoder motors to calculate the position and angle between triangulation operations.

This process is known as odometry. Odometry is more precise for short times and distances. However it has the disadvantage that errors accumulate in the course of time. For this reason the data obtained by odometry is corrected using the data from triangulation.

The soccer robot program has two subroutines for these calculations: the triangulation and odometry subroutines. If you would like, take a closer look at them. But don't be alarmed! They look pretty complicated; and they are.

If everything looks too theoretical for you, don't worry. Simply use the program provided, be amazed at how it works and make a bet with your friends, how many times the robot will have to try before making a goal.

Here are a few more tips on the playing field:

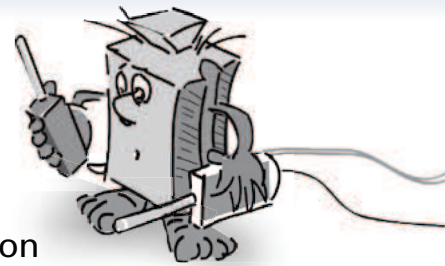
Ensure that a large gap is not present between the floor and the sideboards. Otherwise the camera could think the gap is a stripe and the calculation of the position would be wrong.

To prevent this it is important for you to fold the edges exactly and ensure that the playing field is placed on a level surface. In addition you can also place books or other objects along the outsides of the sideboards to help hold them down.

If the ball continuously rolls toward one corner or frequently stops against one of the sideboards, you can level the playing field by placing flat objects such as newspapers, school notebooks, cardboard, etc. under the corners so the ball will roll back to the middle instead of stopping against the sideboard. This will put more action in to the game!

Trouble Shooting

If things don't work right from the very start there is usually a simple reason. But, it is not always easy to find. That's why we want to give you some information here about possible sources of errors.



Interface Test

Once again here our advice: Check each individual component for proper function with the aid of the ROBO Pro [interface tests](#).

Cables and Wiring

If an electrical component does not work at all, check the cable used to connect it to the [ROBOTICS TXT Controller](#) again. For this purpose use the cable to connect the lens tip lamp to the battery. If the bulb lights up the cable should be okay.

Another source of errors are incorrectly installed plugs (for example, a green plug on a red cable).

Also check that "+" and "-" are correctly connected. For this purpose, compare your model with the illustrations in the assembly instructions.

Loose Connection

A component, which works sometimes and doesn't work at other times, probably has a loose connection somewhere in the wiring for the component.

The most frequent causes for this are:

- Loose Plugs

When the plugs for the cables are too loose, that is wobbly in the jack sockets, they do not provide sufficient contact. In this case you can carefully bend the contact springs apart at the front on the affected connectors using a screwdriver. Just a little, so the plugs are held firmly in the jack sockets when you plug them in.

- Poor Contact between Cable and Plug

Also check the contact between the stripped ends of the cable in the connector and the connector itself. It may be sufficient to tighten the screws in the connector slightly.

Short Circuits

When a positive and negative connection touch one another you have a short circuit. The battery as well as the [ROBOTICS TXT Controller](#) have a built-in fuse to keep them from being damaged by a short circuit. They simply switch off the power supply for a while. Naturally, your model will not work anymore either.

The cause for a short circuit can be either a mistake in the wiring or screws which have not been tightened sufficiently in the connectors. They can touch when the connectors are plugged in, causing a short circuit. That's why you should always completely screw in the screws and plug in the plugs so that the screws cannot touch each other.

Power Supply

Short interruptions or motors, which run too slowly, usually indicate a low battery. In this case, you should charge the rechargeable battery with the battery charger. The battery is completely charged when the red LED on the charger stops flashing and illuminates continuously.

Errors in the Program

Even if no one likes to admit it: Everyone makes mistakes. And especially with complex programs, an error can creep in quite quickly.

If your model still doesn't do what you want it to after you have checked everything on the model itself and have remedied any faults, you should also check your program. Go through it line for line to see if you can find the error.

You can also watch the program running on the monitor in the online mode, which means with the [ROBOTICS TXT Controller](#) connected to the PC. The program element active at any given moment is highlighted allowing you to always see the point where the program is and where the error occurs.

Camera Function

The camera works best when the light is good.

Light too low: It cannot recognize color or motion in the dark.

Remedy: LEDs to illuminate the camera field of vision.

Light too bright: Excessive light - direct sunlight for example - changes the color value and contrast so that the line detector can no longer recognize the lines satisfactorily.

Remedy: Avoid direct sunlight or reduce the light by pulling the drapes or closing the shutters.

Other remedies: Use the software to adapt the camera properties. This can be accomplished with the ROBO Pro software in the property fields for the camera sensors. Here it is possible, for example, to set the sensitivity for recognition of an object and adapt it to the surroundings. Details are given in [ROBO Pro Help](#) Chapter 11.

Last Sources for Help

If, in spite of all this, you have not found the error, there are still two possibilities for obtaining help:

- Email Help

Send an email to fischertechnik describing the problem.

Our email address is: info@fischertechnik.de.

- Internet Help

You can also visit our web site on the Internet at <http://www.fischertechnik.de>. The site includes a forum where you are certain to find help. In addition, you can also become a member of the fischertechnik Fan Club at no charge.

And what else can I do?

- Not finished yet** Was that everything? No, of course not. The experiments and models you tried out in this booklet should only be the beginning. Your "first attempts to walk" so to speak in the gigantic and exciting realm of "ROTBOTICS".
- Fantasy** What we have shown you here is only a very small portion of the possibilities offered by your ROBOTICS TXT Controller and the fischertechnik components. And now it is your turn. You can give your fantasy free rein and simply build what you feel like.
- Changing the present** If you don't have any ideas for your own complete model then just take a look at the models in this booklet. Perhaps something will occur to you how to build a model differently. Or you can change the function of a model.
- Drawing machine** For example you could fasten a pen to the mobile robot, so that it can be moved up and down allowing it to write on a large piece of paper while the robot is moving across it. Then you can draw the figures the robot moves through. This turns your mobile robot in to a drawing machine.
- Racing with friends** You can also use a thick felt-tip pen to draw your own obstacle course on a piece of paper and have the robot move through it. And if your friends have a ROBOTICS TXT Controller, you can do even more interesting things. You can run races to see whose robot can master a given obstacle course faster. And, in addition to allowing your PC to communicate with the ROBOTICS TXT Controller, the Bluetooth interface can also be used to connect a number of computers with one another. Then, for example, you could program two robots, to react to one another. Or dance with each other. A great deal of interesting information on this subject is given in Chapters 4.5 and 7 in the [ROBO Pro Help](#).

Wireless freedom

Does your computer have a WLAN interface? If so you can connect the ROBOTICS TXT Controller to the computer using WLAN instead of a USB cable. If not, you can buy a USB WLAN stick and use it for wireless connection between your ROBOTICS TXT Controller and PC. This is described in the instruction manual for the ROBOTICS TXT Controllers and at <http://www.fischertechnik.de>.

Persistence pays off

So, what are you waiting for? Get started! Invent and experiment! And don't be bothered by little setbacks. Patience and perseverance are primary factors for experimenting. The reward after this is a functioning model.

We hope you have lots of fun trying out your own ideas.

