MODELS 8 TO 10

# Automated guided vehicle 1–4

## Safe navigation in the course

### KEY QUESTIONS:

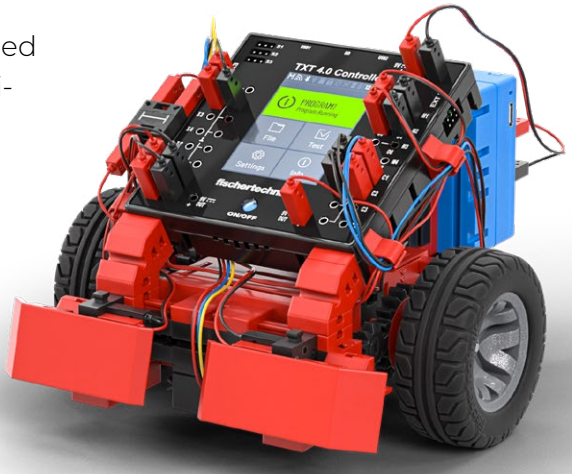
- **Which sensors and control methods are suitable for a automated guided vehicle (AGV) in the course?** ***(Communication)***
- **How can different sensor values (IR sensor, ultrasonic sensor, USB camera) be coordinated for safe navigation?** ***(Collaboration)***
- **What compromises are reasonable between speed, precision, and safety?** ***(Critical thinking)***
- **How can the behavior of the AGV be developed from rule-based to AI-supported control?** ***(Creativity)***

## THE TEACHING CONCEPT AT A GLANCE

| | |
|---|---|
| **Grade level:** | 11–13 |
| **Time required:** | 2–3 double lessons per unit (expandable up to 13 DL) |
| **Degree of difficulty:** | Model  |
| | Programming  up to  |
| **Model type:** | Tabletop models for automated guided vehicles |

## MODEL DESCRIPTION / TASK

The students plan and implement automated guided vehicles (AGV), which they gradually equip with additional sensors and intelligent control. Starting with a base vehicle with encoder motors that performs defined routes and turns as well as initial line following using a track sensor, then advancing to an AGV with an ultrasonic sensor for collision avoidance and a USB camera for interaction with colored surfaces, up to rule loop-controlled or even AI-supported line following, the models develop step by step – and with them the demands on setup, wiring, and programming.




Obstacle detector

The students learn to create state transition diagrams for driving states and to read and use sensor data in a targeted way. From the properties and measurement values of the sensors they determine suitable driving and steering groups, define variables for distance and angle control as well as reaction times for safe course corrections.

From these defined states and parameters, the students first develop rule-based control programs with variables, subprograms, and state logic; then they configure and train a neural network for a regression problem that predicts suitable motor speeds from sensor inputs and is extended through differentiation by the distance sensor. The students test the suitability in driving trials on the course – straight driving, lane keeping, obstacle avoidance, and color surface reactions – and improve their solutions through systematic troubleshooting and speed optimization.

## EVERYDAY RELEVANCE

Sensor-based vehicle control is familiar to the students from technology or computer science lessons and their everyday lives. Familiar applications include cruise control and lane keeping assist in cars, e-scooters with sensors, and robotic vacuum cleaners that follow lines and avoid obstacles.

Embedding in a realistic mobility context creates strong motivation, as students can directly recognize parallels to urban traffic and warehouse logistics.

Integration of the topic into preprofessional orientation is appropriate in automotive engineering, electrical engineering, and robotics/automation, where sensor-driven control and automated actuator control are key competencies.

The combination of sensor technology and AI-supported control is encountered by students not only in industry and traffic but also in the home environment, e.g. in smart home applications, intelligent heating, or automatic lighting – everywhere that measured values trigger decisions and movements are reliably executed.

## SUBJECT REFERENCE

| | |
|---|---|
| **Information technology:** | Advanced programming, conditional loops, functions, state machines, event control, camera integration, P and PD controllers, neural networks, training of a neural network |
| **Physics:** | Movement (distance, time, speed), signal processing, ultrasonic time-of-flight measurement and sound propagation, inertia and braking distance, color recognition, control engineering |
| **Technology:** | Stable construction, construction technology |
| **Mathematics:** | Calculation of terms, scaling, proportionality and linear functions, unit conversion, normalization, regression |

## LESSON PLAN

### Introductory phase

| | |
|---|---|
| Classroom discussion | • Announce the topic.<br>• Ask what the essential features are for controlling automated guided vehicles.<br>• Ask for scenarios in which automated guided vehicles occur.<br>• Discuss possible applications.<br>• Determine requirements for building the models. |
| Support, if necessary | • Show sensors, actuators and components from the assembly kit, use presentation media if necessary. |

### Planning Phase

| | |
|---|---|
| Classroom discussion | • The procedure for building the model and the target function are developed jointly.<br>• The work steps in the app are specified or discussed. |
| Partner or individual work | • The students familiarize themselves with the app and load the corresponding task.<br>• The students define meaningful functions for controlling an AGV using the TXT 4.0 controller.<br>• The students use the app to create the sensor list for the current use case.<br>• The students prepare the course. The driving surface should be as smooth as possible.<br><br>**Note:** The driving experiments should preferably take place either on the floor or on a surface with edge boundaries, in order to avoid the vehicle models falling off. |
| **Optional:** Partner or group work | • The students discuss the sensors that can be used and sketch possible setups.<br>• The students discuss their experiences with automated or autonomous driving in the group. |

## Construction phase for vehicle training (AGV 1)

Partner or individual work

- The students use the app to build the obstacle detector. The app guides them through the program in short steps.
- The students check the wiring of the sensors using the interface test.

## Programming phase for vehicle training (AGV 1)

Partner or individual work

- The students develop a program for simple straight driving as well as for targeted turns with the AGV.
- The students calculate the necessary encoder pulses and make individual adjustments to the conversion factors *"pulses per cm"* and *"pulses per degree"* according to the conditions of the AGV.
- The students extend the AGV with simple obstacle detection and create a simple state transition diagram to analyze the programming task.
- The students develop a program for simple line following using digital control.
- The app guides them step by step with open questions through the programming task.
- The app offers assistance.
- The program is transferred to the TXT 4.0 controller after each differentiation step.

## Experimentation and test phase for vehicle training (AGV 1)

Partner or group work

- The obstacle detector is put into operation.
- Possible malfunctions in the functional sequence must be found and eliminated.
- Potential troubleshooting is possible based on suggestions in the app.
- Possible optimizations in hardware and programming, e.g. adjustments to driving speed and adjustments to the motor speed difference when turning, can be carried out.

## Final/follow-up phase for vehicle training (AGV 1)

**Optional:** Presentation and allocation of differentiations

- Students (or the entire group) eligible for differentiation may be addressed by the teacher. Here, options for avoiding obstacles are discussed.
- The integration of the USB camera using different color spaces is introduced.

## Construction phase for the digital line follower (AGV 2)

Partner or individual work

- The students use the app to build the digital line follower. The app guides them through the program in short steps.
- The students check the wiring of the sensors using the interface test.

Digital line follower

## Programming phase for the digital line follower (AGV 2)

Partner or individual work

- The students sketch a state transition diagram to plan the programming task for obstacle avoidance using the digital line follower.
- The students extend the line-following program from AGV 1 with recognition and avoidance of an obstacle.
- The students plan the integration of the USB camera so that the AGV can react to color surfaces to the right or left of the line.
- The students configure the color recognition in the HSV color space.
- The students define the actions for each color surface and integrate them into the program from AGV 1, so that the AGV now also reacts to recognized color surfaces while line following.
- The app guides them step by step with open questions through the programming task.
- The app offers assistance.
- The program is transferred to the TXT 4.0 controller after each differentiation step.

## Experimentation and test phase for the digital line follower (AGV 2)

Partner or group work

- The digital line follower is put into operation.
- The students test the reactions of the AGV to the color surfaces along the course.
- Possible malfunctions in the functional sequence must be found and eliminated.
- Potential troubleshooting is possible based on suggestions in the app.

## Final/follow-up phase for the digital line follower (AGV 2)

**Optional:** Presentation and allocation of further differentiation

- Students (or the entire group) eligible for differentiation may be addressed by the teacher. Here, the concept of the control loop in control engineering is introduced.

## Construction phase for the analog line follower (AGV 3)

Partner or group work

- The students use the app to build the analog line follower. The app guides them through the program in short steps.


Analog line follower

## Programming phase for the analog line follower (AGV 3)

Partner or group work

- The students plan the integration of the USB camera so that the AGV in this case can follow a line using line recognition with the camera.
- The students configure line recognition with the camera in the RGB color space.
- The students plan and implement a P controller.
- The students extend the programming to a PD controller.
- The app guides them step by step with open questions through the programming task.
- The app offers assistance.
- The program is transferred to the TXT 4.0 controller after each differentiation step.

## Experimentation and test phase for the analog line follower (AGV 3)

Partner or group work

- The analog line follower is put into operation.
- Possible malfunctions in the functional sequence must be found and eliminated.
- Possible troubleshooting is possible based on suggestions in the app.
- The students test different settings for the proportionality factor of the P controller.
- The students compare the driving behavior of the AGV when controlled with a P vs. a PD controller.

## Final phase for the analog line follower (AGV 3)

**Optional:** Presentation and allocation of further differentiation

- Students (or the entire group) eligible for differentiation may be addressed by the teacher. Here, the concepts of neural networks and AI are to be presented.

## Construction phase of the AI line follower (AGV 4)

Partner or individual work

- The students use the app to build the digital line follower. The app guides them through the program in short steps.
- The students check the wiring of the sensors using the interface test.
- If the digital line follower in AGV 2 has already been built, the construction phase is omitted.

## Programming phase of the AI line follower (AGV 4)

Partner or individual work

- The students familiarize themselves with the structure of a simple neural network.
- The students configure a neural network using the app, which calculates motor speeds from the track sensor data and provide training data.
- The students extend the programming from AGV 1 if necessary, so that the motor speeds of the two motors are now determined by the neural network.
- The students train the neural network.
- The students supplement the program with the query of the distance sensor.
- The students extend the neural network with an input neuron that receives the sensor data from the distance sensor.

## Experimentation and test phase of the AI line follower (AGV 4)

Partner or group work

- The students observe the driving behavior of the AGV and test how adjustments to the training data table, different training settings, and different configurations of the neural network affect driving behavior.
- The students test the braking behavior of the AGV.
- The students optimize the driving and braking behavior of the AGV.

## Final phase of the AI line follower (AGV 4)

**Optional:** Presentation and allocation of further differentiation

- Particularly efficient students or groups can be given the extension task of developing their own use cases for the driving behavior of the AGV and selecting the necessary sensors for this.
- The students can combine the programming knowledge and control techniques acquired during the lessons to develop an AGV with the most consistent driving behavior possible and a wide range of reactions.
- Different groups can compete against each other: Which AGV completes the course the fastest? Which AGV is the most stable against disturbances? Independent work without learning cards is intended here.

Discussion in plenary

- Project debriefing in class.
- Clarification of future possible applications in everyday life

## METHODOLOGICAL AND INSTRUCTIVE TIPS

**Differentiation options**

Depending on the duration of the lesson series and the abilities of the students,

- the complexity of the task can be increased by incorporating multiple sensors.
- an extension of the AGV's capabilities through more complex strategies for avoiding obstacles or capturing and reacting to color stimuli from the environment can be implemented.
- the driving behavior of the AGV can be made more consistent through the use of P and PD controllers.
- the motor control for line following or braking can be taken over by a neural network.

**Motivational aspects**

Working with automated guided vehicles (AGV) ties directly to students' everyday experiences. Already during **vehicle training** they experience how just a few lines of code can reliably set a vehicle in motion – direct feedback that is highly motivating and sparks curiosity.

With the **digital line follower**, the everyday reference is reinforced by parallels to robotic vacuum cleaners, robotic lawnmowers, or driver assistance systems that recognize lines and independently avoid obstacles. The idea that the AGV can now "see" and react like a robot promotes identification with the task and increases interest in practical testing.

The **analog line follower** provides exciting insights through camera use and control engineering into modern vehicle technologies such as lane keeping assist systems. Here it becomes clear how parameter changes or the addition of a derivative component influence driving behavior – a motivating field for experimentation.

With the **AI line follower**, artificial intelligence finally takes center stage: Students experience how training a neural network improves driving behavior. This bridges to current discussions about AI in everyday life and gives students the opportunity to understand and apply future technologies themselves.

## BASICS OF CONTROL ENGINEERING

A P controller is the simplest form of **feedback control**: It continuously compares a desired setpoint (target) with the current measured value (actual) and converts the resulting deviation – the error – directly into a correction whose size is **proportional** to the error. The larger the error, the stronger the counteraction; the "sensitivity" of the system is determined by the proportional factor $k_P$. If $k_P$ is too small, the system reacts sluggishly and returns to course only slowly; if it is too large, it overreacts, begins to oscillate, and becomes unstable. In the classroom context of the analog line follower, this means: The camera measures the lateral deviation from the line, the P controller adjusts the left and right motor speed in opposite directions and brings the vehicle back to the center of the line. Th P controller is thus the basis of any further refinement (e.g. through a D component) and is ideally suited to clearly demonstrate the basic principle of automatic, stabilizing controls with little programming effort.

The central control variable in line following is the lateral deviation of the line center from the image center of the camera. We denote this deviation as error $e$ in **pixels** (positive if the line is to the right of the image center; negative if it is to the left). The controller generates a control variable $u$ from this error, with which the left and right motor speeds are adjusted in opposite directions. With a base speed $v_0$ and the limitation to the permissible range $[0, v_{max}]$, the result is

$$v_{left} = \text{clip}\,(v_0 + u, 0, v_{max}),\ \ v_{right} = \text{clip}\,(v_0 - u, 0, v_{max})\,.$$

Here, the clip function limits ("saturates") the speed value to the permissible range. If the line is to the right of the center, the AGV should turn right to find the line again. To do this, the right wheel must turn slower and the left wheel faster. If the line is to the left of the center, the AGV must make a left turn, i.e. the left wheel must now turn slower and the right wheel faster.

With the **proportional controller (P controller)**, the control variable is proportional to the current error

$$u = k_P \cdot e,$$

where $k_P$ has the unit "speed per pixel."

**Example 1:** $v_0 = 320$, $v_{max} = 500$, $k_P = 1.2$, line at +40 px, to the right of the image center, thus $e = +40$ px. This results in $u = 1.20 \cdot 40 = 48$, $v_{left} = 368$, $v_{right} = 272$ (no clipping necessary).

**Example 2:** $v_0 = 320$, $v_{max} = 500$, $k_P = 1.2$, line at −35 px, to the left of the image center, thus $e = -35$ px. This results in $u = 1.20 \cdot (-35) = -42$, $v_{left} = 278$, $v_{right} = 362$ (no clipping necessary).

If a speed exceeds the limits $[0, v_{max}]$, it is limited by the clip function. However, permanent clipping should be avoided by using slightly smaller values for $k_P$ or $v_0$.

The **proportional-derivative controller (PD controller)** extends the P controller with a predictive D component, which considers the rate of change of the error. With discretization $i$ formulated, at a sampling time $\Delta t$ for the control variable $u[i]$

$$u[i] = k_P \cdot e[i] + k_D \frac{e[i] - e[i-1]}{\Delta t}\,.$$

Here, $k_D$ has the unit "speed per pixel per second." As the AGV approaches the line center, the D component acts as a brake and reduces overshooting. If the error increases, i.e. the AGV deviates further from the line, the D component reinforces the correction and thus helps return to the line more quickly.

**Example 3 (line right, approaching the center):**
$v_0 = 320$, $v_{max} = 500$, $k_P = 1.2$, $k_D = 0.03$ (speed per pixel per second),
$\Delta t = 0.02\,\text{s}$ (sampling rate), $e[i] = +45$ px, $e[i-1] = +60$ px.
For the P component then $k_P \cdot e = 1.20 \cdot 45 = 54$.

The rate of change is $(45–60)/0.02 = –750$ px/s. The D component thus provides $0.03 \cdot (–750) = –22.5$. Overall, this results in $u = 54 – 22.5 = 31.5$. Thus, the motor speeds become $v_{left} = 320 + 31.5 = 351.5$, $v_{right} = 320 – 31.5 = 288.5$. Without the D component, a pure P controller would give $u = 54$, i.e. $v_{left} = 374$, $v_{right} = 266$; the PD component visibly damps the correction because the vehicle is already approaching the line center.

**Example 4 (line left, error increasing):**
$v_0 = 320$, $v_{max} = 500$, $k_P = 1.2$, $k_D = 0.03$ (speed per pixel per second),
$\Delta t = 0.02$ s (sampling rate), $e[i] = –45$ px, $e[i – 1] = –20$ px.
For the P component then $k_P \cdot e = 1.20 \cdot (–45) = –54$.

The rate of change is $(–45–(–20))/0.02 = –1250$ px/s. The D component thus provides $0.03 \cdot (–1250) = –37.5$. Overall, this results in $u = –54 – 37.5 = –91.5$. Thus, the motor speeds become $v_{left} = 320 – 91.5 = 228.5$, $v_{right} = 320 – (–91.5) = 411.5$; the PD component significantly reinforces the correction because the vehicle is moving further away from the line center.

If the program is to work for different image widths W, it would make sense to use a normalized error, i.e. $e_n = x/W \in [–1, 1]$.

## PROGRAMMING SKILLS

- Program start
- Infinite loop **repeat forever**
- Camera configuration
- Loop **repeat while**
- Command **wait until**
- Condition **if – do**
- Use of variables and their change
- Working with variables
- Working with subprograms
- Programming of neural networks
- Input of AI training data

**Optional download:**
- Circuit diagram
- Building instructions

## ADDITIONAL MATERIALS

- Drawing media (paper, whiteboard, or projection screen).

## FUNCTIONS OF THE MODEL AND THEIR TECHNICAL SOLUTIONS

| Function of the sensors/actuators | Technical solution |
|---|---|
| Rotation of encoder motors | Speed adjustment for vehicle control |
| Mini push button | Obstacle detection **(AGV 1)** |
| Measurement of brightness differences | Line recognition **(AGV 1–2, AGV 4)** |
| Measurement of distances | Collision avoidance **(AGV 2, AGV 4)** |
| **AGV 2:** Color recognition using USB camera | Reaction to color surfaces |
| **AGV 3:** Line recognition using USB camera | Line following using a P controller and a PD controller |
| **AGV 4:** Line recognition using track sensor | • Development of a neural network<br>• Input of training data<br>• Implementation of a regression problem using AI |
| **Further differentiation options** | Optimization of speed control, optimization of obstacle avoidance strategies, optimization of the strategy for rediscovering the line after complete line loss, design of a custom vehicle with self-selected sensor configuration |

## MATERIAL LIST

| Sensors | Function |
|---|---|
| 1 On/Off push button on the TXT 4.0 controller | Switching on the AGV |
| 2 mini push buttons | Obstacle detection **(AGV 1)** |
| 1 track sensor (with 2 IR sensors) | Line recognition **(AGV 1–2, AGV 4)** |
| 1 ultrasonic sensor | Distance measurement **(AGV 2, AGV 4)** |
| 1 USB camera | Color recognition **(AGV 2)**<br>Line recognition **(AGV 3)** |

| Actuators | Function |
|---|---|
| 2 encoder motors | Vehicle drive |
| 2 LEDs (2 × white) | Headlights **(AGV 2–4)** |