Name: \_\_\_\_\_\_\_\_\_\_ Class: \_\_\_ Date: \_\_\_\_\_\_\_\_

# Task 2

# Brake assistant, cruise control and lane departure warning system

*Braking in a timely fashion before an obstacle, and independently staying on track and maintaining speed are important driver assistance systems which will be developed in this task. To ensure the vehicle does not drive off in an uncontrolled manner during the test, it will be equipped first with an “emergency stop” function.*

## Construction task

For this task, we will need the servo motor, the ultrasound sensor and the USB camera. Connect the ultrasound sensor to I1, the servo motor to S1 and the USB camera to USB1 (see the wiring diagram). Ensure correct polarity for the servo plug: the brown cable on the left, and the orange one on the right.

**Important note**: When the TXT is started, the servo automatically switches to “forward setting”. Insert the servo lever so that both front wheels are aligned roughly exactly forward in this position.

**Note**: If you move the servo manually with the TXT switched on, you may damage it. The servo should also not approach any position that is beyond the stop of the servo lever – only use positions between around 100 and 400 for the servo commands. The steering of our vehicle is a little smaller than this; it is around servo values 130 to 370.

You can very easily measure the steering angle range within which the front wheels can still turn freely using the interface test.

Now, we will equip the vehicle with a brake assistant, a speed control and a lane departure warning system.

## Programming tasks

**1. Emergency stop**

We will start with an “emergency stop” mechanism before we have the vehicle drive autonomously: When the vehicle hears a loud clap, it should stop immediately.

Program the “emergency stop” function as a concurrent process (thread) controlled via the microphone on the camera. You can determine an appropriate volume threshold at which the motor should stop by experimenting; display the volume value determined by the microphone on the console.

**Please note**: The motor itself produces a relatively loud noise.

**2. Forward travel**

You will see that the vehicle does not drive exactly straight ahead. Before we automatically align steering to the track in the experimental tasks, you should first attempt to set the servo so that the vehicle travels roughly straight ahead.

You can approximate this initially using the interface test. However, you will have to test the results through experimenting: Write a Blockly test program and correct the value for the initial servo setting, until the vehicle maintains its direction fairly precisely across a test track of 2 m.

In the following programs, specify the correct servo setting in a variable, and align the servo using this value at the start of the program.

**3. Braking assistant**

A braking assistance system should now ensure that your autonomous vehicle does not run into a stationary (or moving) obstacle if the drive is inattentive or reacts too slowly. To do so, it should initiate braking promptly so that the vehicle doesn’t come within 10 cm of the obstacle.

You can measure the distance from an obstacle (or a vehicle driving in front of your own) using the ultrasound sensor. You are already familiar with it from the Robotics TXT 4.0 Base Set.

Write a Blockly program to do so, and test it out with different obstacles. Use the “emergency stop” thread from programming task 1 to stop the vehicle if necessary by clapping.

**Please note**: Your vehicle will also travel a certain distance during the time it takes to measure the distance from the obstacle.

**4. Speed control**

One driver assistance system already included in most mid-range vehicles is a *speed control*: This is also called a cruise control, and ensures that the vehicle maintains a specified speed [1].

First, review the following illustration of a control circuit. You are already familiar with it from task 8 from the Robotics TXT 4.0 Base Set.



Reference variable *w*

Manipulated variable *u*

Disturbance variable *d*

Controlled system

Controller

Deviation *e*

Control variable *y*

4a. What variables of the speed control correspond to the parameters *w*, *e*, *u*, *d* and *y*? Label the following control circuit accordingly:



Controlled system

Controller

4b. Program a speed controller in Blockly. The desired speed must be specified in m/h in one variable as a command variable.

**Tip**: You can use the “map” command to convert the reference variable into the required motor voltage very elegantly, if you know the vehicle speed in m/h.

**Please note**: The speed control may not increase the speed once again following an “emergency stop” or stop in front of an obstacle. Adjust these two threads in your program accordingly.

## Experimental task

**1. Lane departure warning with P controller**

Now, the vehicle should learn to independently follow a track. It should orient its travel along the right edge of the track, with the help of the camera. You should also implement this driver assistance system using a proportional controller (P controller).

1a. First, label the following control circuit:



Controlled system

Controller

1b. Now, add a proportional controller to your main program that corrects the servo steering depending on the lane deviation.

**Tip**: Configure the line detection of the camera so that it delivers the position “0” when the vehicle is standing parallel to the right edge of the track at a distance of 2 cm.

**Tip**: Limit the speed of your vehicle to 350 and test the P controller initially with a very small value ($k\_{p}=0.1$). Increase the proportionality factor until the controller “starts up” and then choose the value at which the controller “engages” most quickly.

Use the straight track section on the enclosed sheet to test your controller.

1c. Add a text output to your program that indicates

* the time (in ms) which has passed since the start of the program and
* the value for the current position

separated by a space on the console after each change in the position of the detected track boundary.

After each test run with a different value for $k\_{p}$, copy the information output on the console into a spreadsheet, and display the values graphically in a diagram (x: Time, y: Position). Choose the proportionality factor $k\_{p}$, at which the curve progression “engages” most quickly.

**2. Lane departure warning with PD controller**

Just like the buggy in task 8 of the Robotics TXT 4.0 Base Set, you can add a “D” factor (differential factor) to the controller that takes the amount of change in deviation from the detected track boundary from the position “0” into account in the steering correction, to further dampen overshooting.

2a. Expand the P controller from experimental task 1 so that it is a PD controller.

2b. Complete test runs with different values for the differential factor $k\_{d}$ and use a spreadsheet program to display the data graphically.

**Tip**: Start your tests with the differential factor $k\_{d}=0.01$ and increase its value in increments of 0.005 until the overshoot of the P controller is dampened well.

2c. If you place multiple straight and curved track sections in a line with one another, you can test the lane departure warning assistant on challenging courses.

Annex

# Task 2: Brake assistant, cruise control and lane departure warning system

## Required materials

* PC for program development, local or via web interface.
* USB cable or BLE or WiFi connection to transmit the program to the TXT4.0.
* Travel route with markings on the enclosed sheet (or as a print out of the travel route file)
* Obstacle (such as a book or a box)

## Further information

[1] Jim Meininghaus: [*Die Geschichte des Tempomaten. Wie ein Blinder das Autofahren veränderte*](https://www.motor-talk.de/news/wie-ein-blinder-das-autofahren-veraenderte-t4865108.html). 03.03.2014, motor-talk.de.

[2] Thomas Paulsen: [*Autonomes Fahren: Die 5 Stufen zum selbstfahrenden Auto*](https://www.adac.de/rund-ums-fahrzeug/ausstattung-technik-zubehoer/autonomes-fahren/grundlagen/autonomes-fahren-5-stufen/). 07.11.2018, adac.de.

[3] Online diagram editor for creating state diagrams (Format drawio): <https://www.diagrammeditor.de/>