­­­LAB4

The goal of this laboratory was to combine the line sensor from Lab 1 and the motor controller from Lab .. to make the Turtlebot follow a given line autonomously.

The infrared line sensor:

We are using a Pololu QTR reflectance sensor which can detect how much infrared light has been reflected from a surface. With a certain threshold we can than determine if a surface is dark or light. The installed sensor is located on the lower front of the robot. It is about 10 cm wide and consists of an array of 8 photo diodes with each one photo transistor. The distance between every sensor is 8 mm. This gives us a total of 8 individual sensors that can differentiate between dark and light areas. This data is retrieved by the controller MCU through a port expander board using a I2C bus connection. The sensor data encoded in a binary format and received as an integer. To retrieve the original data, we have to convert the integer back to a binary number. This binary number has 8 bits, each storing the value of one sensor.

Properties of the line:

The line is assumed to be wide enough to trigger at least one of the infrared sensors and at most two at the same time. The goal is to create a controller algorithm that keeps the line between the two middle sensors 3 and 4. We are implementing a tracking error and set this position to tracking error 0. We then define the tracking error to be greater than zero when the line is on the left side of the sensor. If the tracking error is less than zero, the line is on the right side of the sensor.

Function

Ein Bild, das Text, Screenshot, Schrift, Software enthält.

Automatisch generierte Beschreibung

The calc\_error\_line function does the calculation of the line error. The input of this function is an 8-dimensional array, containing the state of each infrared sensor coded in binary. A logical one means, that the corresponding sensor has detected a dark spot. In the for loop we iterate through every value in the binary[] array, ergo through all sensors data. We calculate the sum of the binary[] array to get the total number of triggered sensors. Furthermore, we are filling the distance\_from\_middle array with the opposing distance of every sensor from the middle. The distance between every sensor is 4mm. These values get negative to indicate a distance to the left and positive to indicate a distance to the right. The sum\_dist value is the sum of every distance from the center if the sensor was activated. Calculating the line\_error by dividing the sum of the distance of each activated sensor with the total number of activated sensors. This line\_ error ten represents the offset of the black line from the center, regardless of how wide the line is. Even if only one sensor is not activated, we still get a useful line error value.

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

The calc\_yaw\_error() function calculates a much stabler result for our error value. It includes the wheelbase dimension and the distance between the line sensor and wheels. This determines the position of the turning point of the bot, which is between the two wheels, shown in figure …



D = 165 mm

H = 85 mm

r = 34 mm

The main structure of our line following code:



Calulating the reference for the motors:

Based on the yaw error, we can now calculate our reference for the two motors. Since our motor controller is designed to compute rotations per minute, we set a base speed of 100 rpm and depending on the direction of the motor add or subtract our yaw error. To give the yaw error more effect on the base speed, we incorporated an additional gain. Testing different gains, we first found a relay able solution with a base speed of 100 rpm +- yaw\_error \* 12. To furthermore improve the performance a linear or even an exponential gain was necessary. This would increase the impact of the yaw error in tighter corners. This would also make sure to decrease the gain when the line is going straight. The robot would be very stable on straight lines and keep its agile performance in sharp corners.

A very simple solution was to square the yaw error. This would increase the gain exponentially. To keep the sign of the yaw error, the absolute value of the yaw error was multiplied by itself, see figure …

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung