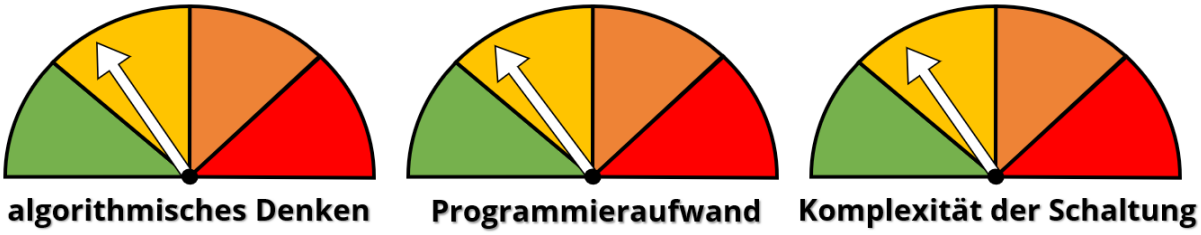
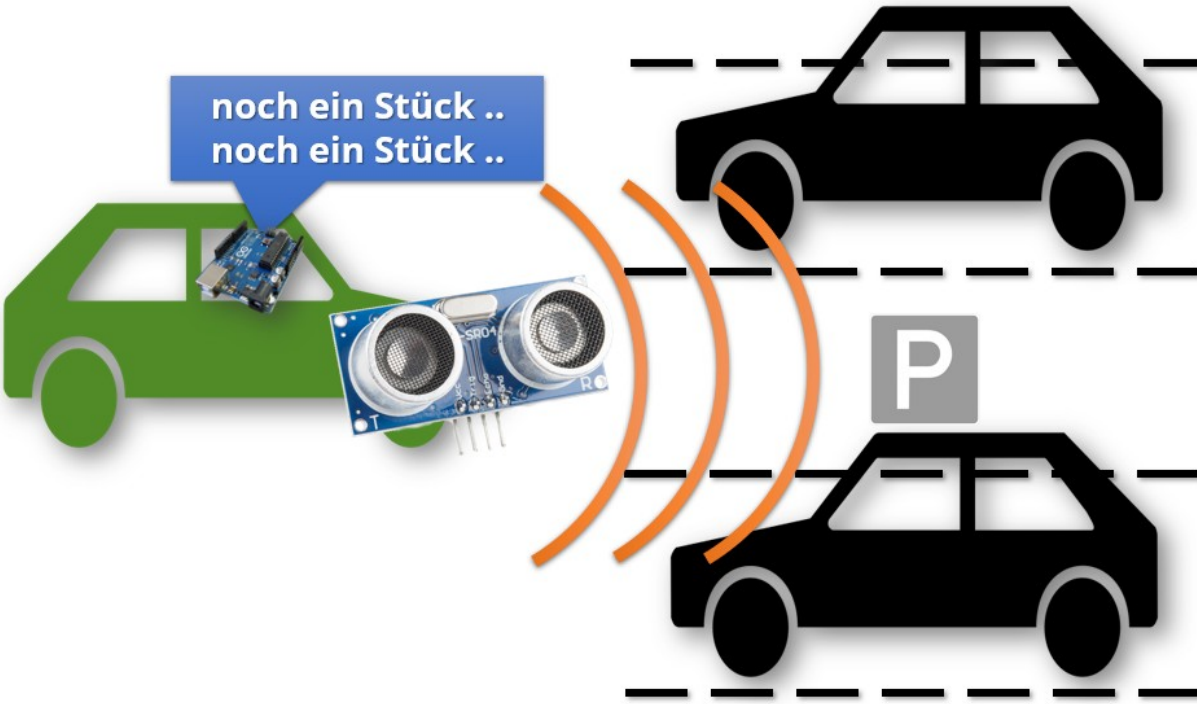
TECHNISCHE UNIVERSITÄT DRESDEN

Arduino Uno

Station 7b | Parksensor - Infrarotsensor





Professur für Didaktik der Informatik

SYLBER Synergetische Lehrerbildung im exzellenten Rahmen

Das Maßnahmenpaket „TUD-Sylber² – Synergetische Lehrerbildung im exzellenten Rahmen“ wird im Rahmen der gemeinsamen „Qualitätsoffensive Lehrerbildung“ von Bund und Ländern aus Mitteln des Bundesministeriums für Bildung und Forschung gefördert.

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

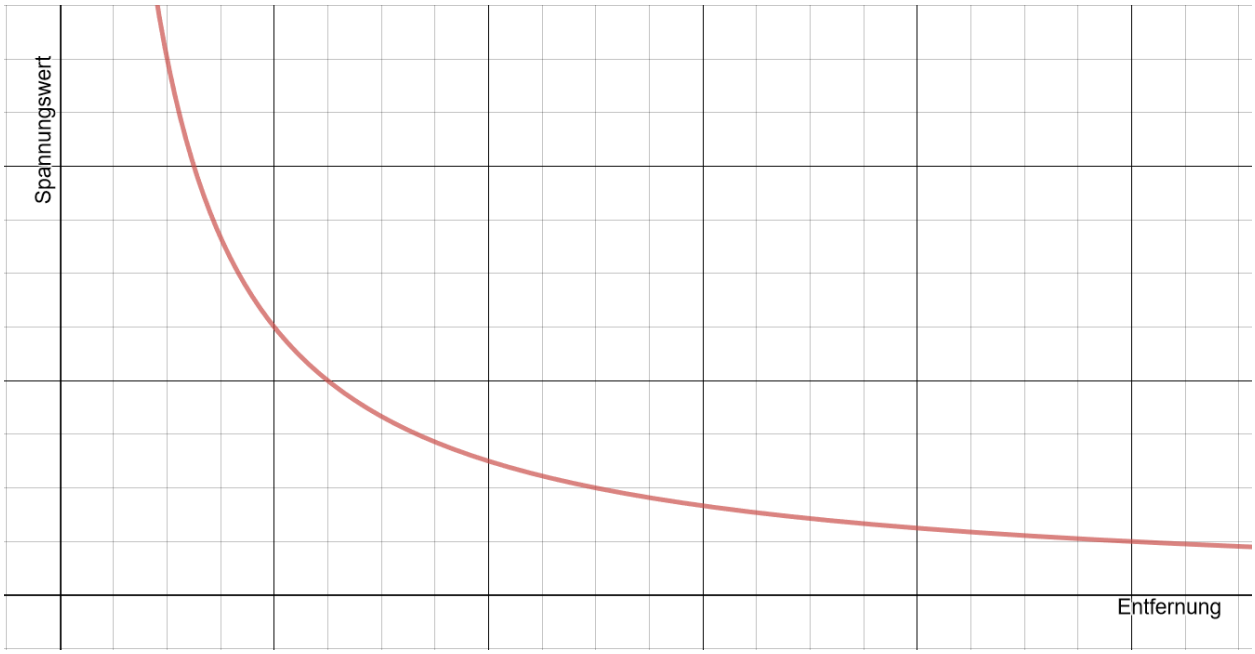
ZIEL DER STATION

DEINE AUFGABE

Du hast dich für den Infrarotsensor entschieden! Mit diesem werden wir jetzt Schritt für Schritt einen Rückfahrsensor entwickeln. Dazu schauen wir uns erst einmal an wie dieser funktioniert und angeschlossen wird. Im Anschluss werden wir den dazugehörigen Programmcode schreiben und den Abstand zu einem Gegenstand bestimmen.

DIE FUNKTIONSWEISE DES SENSORS

Als Output liefert der Infrarotsensor ein analoges Spannungssignal. Dies ist invers proportional zur Entfernung, dass heißt: Je weiter der Gegenstand entfernt ist, desto kleiner ist der Spannungswert. Dabei handelt es sich hierbei nicht um konkrete Entfernungswerte, sondern nur um relative Werte, welche ein Verhältnis angeben. Im Diagramm sieht dies dann wie folgt aus:



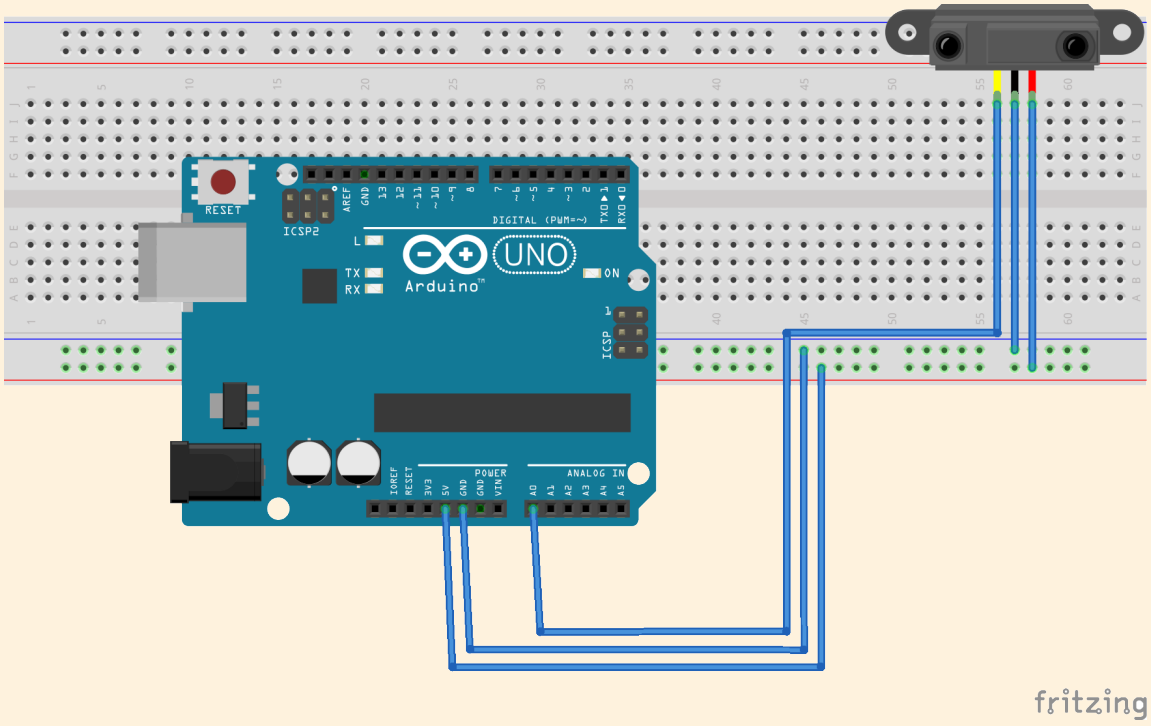
Aufgrund dessen, dass der Sensor einen Wertebereich von 10 bis 80 Zentimeter hat, liefert dieser außerhalb des Bereichs keine verlässlichen Werte, dies gilt im Anschluss dementsprechend abzufangen.

ANSCHLIESSEN DES SENSORS

Der Sensor hat 3 Kabel. Dabei ist sind die roten und schwarzen Kabel für die Spannungsversorgung da. Das rote Kabel wird mit `+5 V` verbunden und das Schwarze mit `GND`. Das weiße Kabel liefert uns den bereits benannten analogen Spannungswert und wird daher mit einem analogen Pin verbunden.

AUFGABE 1 – VERBINDE DEN SENSOR

Beginne zunächst damit, den Sensor mit dem Arduino zu verbinden. Nochmals gezeigt wird dir dies in der untenstehenden Abbildung.



ERSTE WERTE AUSLESEN

Beginnen wir zunächst damit ein kleines Programm zu schreiben, um die Werte des Sensors auszulesen. Dazu legen wir uns wieder eine Variable an, welche die Nummer des Pins speichert, an dem der Sensor angeschlossen ist. Weiterhin benötigen wir eine Variable um den Abstandswert zu speichern.

```
int infrarotsensor = A0;
int abstand = 0;
```

Wir wollen weiterhin die Werte im seriellen Monitor ausgeben, somit müssen wir diesen innerhalb der `setup()`-Methode starten mit: `Serial.begin(9600);`. Innerhalb der `loop()`-Methode lesen wir jetzt den analogen Spannungswert aus und geben diesen im Anschluss im seriellen Monitor aus.

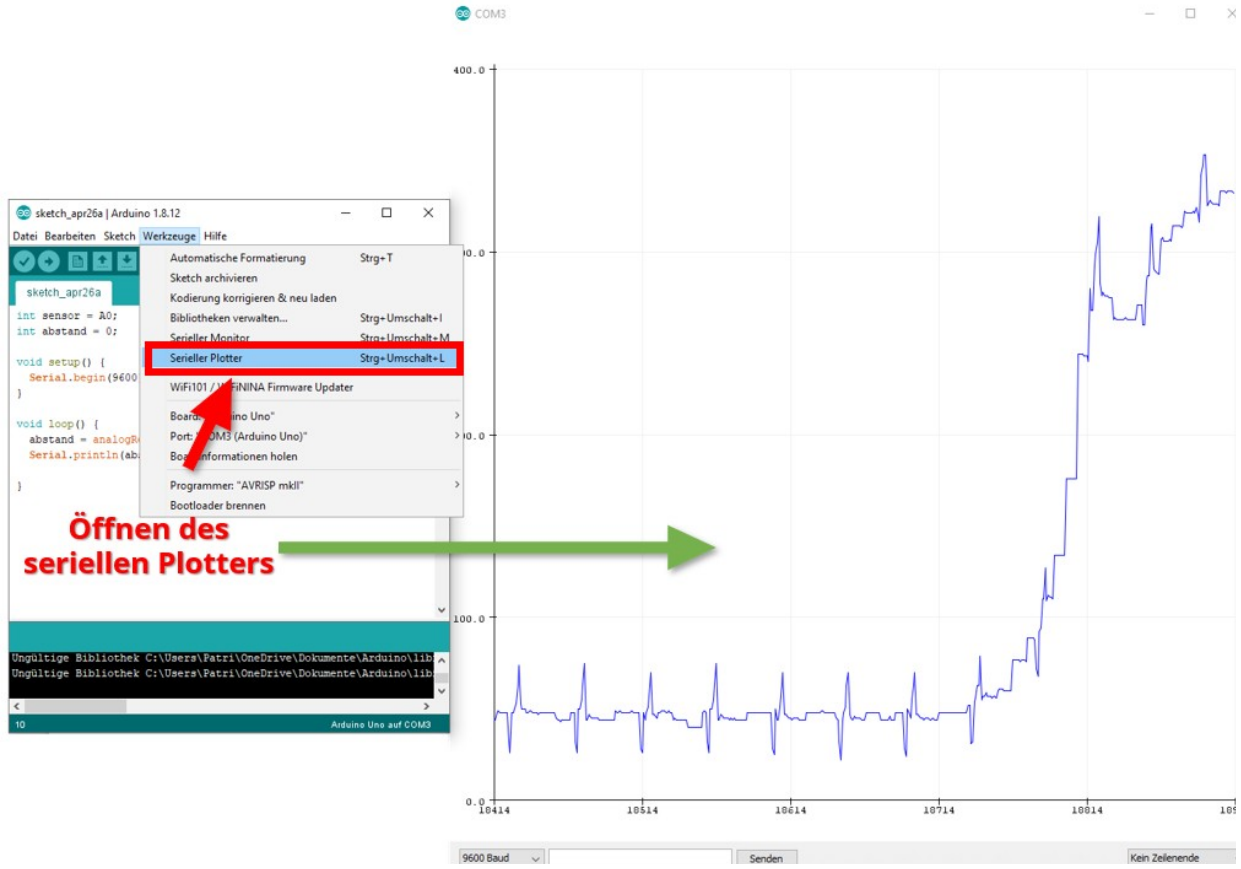
```
abstand = analogRead(infrarotsensor);
Serial.print(abstand);
```

AUFGABE 2 – WERTE ANZEIGEN LASSEN

Öffne einen neuen Sketch und schreibe die oben beschrieben Codezeilen. Übertrage dann dein Programm. Schaue dir im dir bereits bekannten seriellen Monitor die Werte an. Variiere den Abstand zu einem Hindernis, beobachte die Werte und notiere deine Erkenntnis.

_____

Neben den reinen Werten, wollen wir uns die Werte einmal grafisch in einem Diagramm anzeigen lassen. Schließe dazu den seriellen Monitor und öffne den seriellen Plotter, wie dir dies unten gezeigt wird.



AUFGABE 3 – WERTE PLOTTEN

Lasse dir nun die Werte im Diagramm anzeigen. Variiere wieder den Abstand und beobachte das Verhalten des Graphen. Was kannst du für ein Verhalten im Wertebereich des Sensors (10 bis 80 cm) feststellen?

_____

Was passiert wenn der Wertebereich unter- oder überschritten wird?

_____

STARK SCHWANKENDE WERTE

Wie dir mit Sicherheit aufgefallen ist, kannst du dir noch so viel Mühe geben den Sensor absolut still zu halten und trotzdem schwanken die Werte sehr stark. Aus diesem Grund werden wir jetzt einen Durchschnittswert ermitteln, dass heißt wir messen fünf mal mit kurzen Pausen dazwischen und berechnen daraus den Durchschnitt.

Wie machen wir das? Wir benötigen insgesamt 5 Variablen für die jeweiligen Messungen. Wir ändern also unsere bereits existierende Variable

```
int abstand = 0;
```

ab zu

```
int abstand1 = 0;
```

und ergänzen 4 weitere, welche wir einfach durchnummerieren. Danach bestimmen wir innerhalb der `loop()`-Methode 5-mal den analogen Spannungswert des Sensors, mit kurzen Pausen dazwischen.

```
abstandsWert1=analogRead(Infrarotsensor);
delay(20);
```

Jetzt fehlt quasi nur noch der Durchschnitt. Dazu addieren wir alle gemessenen Werte und teilen die Summe durch die Anzahl der Messung. Auch dazu benötigen wir wieder eine neue Variable zum Speichern des Ergebnisses. Da wir diesmal dividieren wollen und nicht mit Sicherheit eine Ganzzahl herauskommt, benötigen wir den Datentyp `double`.

Für die Variable ergibt sich also:

```
double durchschnitt = 0.0;
```

und für die Durchschnittsberechnung:

```
durchschnitt =(abstandsWert1 + abstandsWert2 + abstandsWert3
                + abstandsWert4 + abstandsWert5) / 5.0;
```

ACHTUNG

Zunächst musst du darauf achten, dass die Programmiersprache keine Kommas kennt bei gebrochenen Zahlen. Dies wird mit Hilfe eines Punktes repräsentiert. Auch das dir bekannte Zeichen für „geteilt-durch“ suchst du vergebens auf deiner Tastatur, im Code wird dies einfach mit Hilfe eines Schrägstriches (im engl. slash) „/“ ausgedrückt.

Jetzt musst du nur die Variable bei der Ausgabe im seriellen Monitor abändern.

AUFGABE 4 – DURCHSCHNITTSWERT

Ergänze in deinem Programmcode die nun notwendig gewordenen neuen Variablen, füge die 5 Messungen hinzu und berechne den Durchschnitt.

Die Werte sollten nun deutlich präziser werden und weniger schwanken. Das gilt es natürlich auszutesten – nutze dazu den seriellen Monitor oder Plotter.

BREAKPOINTS FÜR DIE FUNKTIONALITÄT DES ABSTANDSSENSORS

Um unsere Abstandsmessung als Parksensor zu verwenden, müssen wir sogenannte Breakpoints festlegen. Damit ist es uns möglich in Abhängigkeit der Entfernung spezifische Warnungen ausgeben zu lassen oder auch fehlerhafte Werte abzufangen. Da der Sensor allerdings nicht linear arbeitet, müssen wir zunächst Werte bestimmen.

AUFGABE 4 – SENSORWERT VS. ABSTAND

Bestimme mit deinem jetzigem Programm, dem seriellen Monitor und einem Lineal die jeweiligen Abstände zum Sensor. Gehe dabei in 10 Zentimeter Schritten vor und notiere in der untenstehenden Tabelle die Werte des Infrarotsensors.

Abstand:	10 cm	20 cm	30 cm	40 cm	50 cm	60 cm	70 cm	80 cm
Sensorwert:								

Mit diesen Werte können wir jetzt unsere Warnungen festlegen. Um dies zu realisieren nutzen wir die **einfache Verzweigung** und die logischen Vergleichsoperatoren **UND** „**&&**“, sowie **ODER** „**||**“. Zu Beginn wollen wir erstmal fehlerhafte Werte abfangen, dazu zählt alles was über oder unter dem Wertebereich des Sensors liegt. Sprich alle Entfernung unter 10 Zentimetern und über 80 Zentimeter. Im Code sieht dies dann wie folgt aus:

```
if (durchschnitt < _________ ||durchschnitt > _________){
    Serial.println("fehlerhafte Messung – kein Ergebnis");
}
```

Schreibe auf die Striche deine Sensorwerte für den jeweiligen Abstand.

Weitere Regeln kannst du mit weiteren Verzweigungen und Bedingungen festlegen mit Hilfe von:

```
else if(durchschnitt <= _______ && durchschnitt >= _______){
     Serial.println( … deine Warnung … );
}
```

AUFGABE 5 – BREAKPOINTS FESTLEGEN

Ergänze deinen Code um die oben benannte Verzweigung. Ergänze danach weitere Punkte mit spezifischen Warnungen. Diese sollten ungefähr der Realität entsprechen – je näher man dem Hindernis kommt, desto ausdrucksvoller muss die Warnung sein. Teste mit einem Hindernis in verschiedenen Entfernungen zum Sensor und überprüfe mit einem Lineal.

DU HAST NOCH NICHT GENUG?

Es gibt noch eine Zusatzaufgabe zur Station. Dort wird es darum gehen, dass textuelle Feedback zu ersetzen. Denn für einen Einsatz im Auto sind natürlich nur optische und akustische Warnungen geeignet! - Interesse? Dann frage die Betreuer nach dem Zusatzblatt!
