Üben mit Buchstaben (Lehrkräfte)

Arbeitsblatt 1

Aufgabenstellung:

Findet heraus, welcher Buchstabe entsteht, wenn der Rover die folgenden Anweisungen ausführt:

1. Fahre 2 Einheiten geradeaus.
2. Drehe dich um 90° nach rechts.
3. Fahre eine Einheit geradeaus.
4. Drehe dich erneut um 90° nach rechts.
5. Fahre 2 Einheiten geradeaus.
6. Fahre eine Einheit zurück.
7. Drehe dich um 90° nach rechts.
8. Fahre eine Einheit gerade aus.

! Gebt zuerst eine Vermutung ab und notiert diese.

Übersetzt die Anweisungen in ein Programm am Rover.

Bringt einen Stift am Rover an und lasst ihn die Fahrt aufzeichnen. Hattet ihr Recht?

Lasst den Rover nun einen anderen Buchstaben eurer Wahl fahren.

Schafft ihr es, den Rover ein ganzes Wort schreiben zu lassen?

Mögliche Lösung:

*buchstabe.py 1/16

```
# Rover Coding
#================================
import ti_rover as rv
from math import *
import ti_plotlib as plt
from ti_system import *
from time import *
#================================
rv.forward(2)
rv.right(90)
rv.forward(1)
rv.right(90)
rv.forward(2)
rv.backward(1)
rv.right(90)
rv.forward(1)
```

Es handelt sich um den **Buchstaben A.**

Kommentar für Lehrkräfte:
- Für jede Schüler:innengruppe ein Plakat am Boden festkleben.
- Stifte für die Aufzeichnung des Weges zur Verfügung stellen.

Vielecke (Lehrkräfte)
Arbeitsblatt 2

Aufgabenstellung:

Der folgende Code lässt den Rover ein regelmäßiges Sechseck fahren.

```
for i in range(6):
    rv.forward(2)
    rv.left(60)
```

Erstellt das Programm für das Sechseck am Rover. Bringt einen Stift am Rover an und lasst ihn die Fahrt aufzeichnen.

Was stellt ihr fest?

Erstellt nun ein Programm, welches den Rover ein regelmäßiges Achteck fahren lässt.

Lasst den Rover ein weiteres regelmäßiges Vieleck eurer Wahl fahren.

Wie habt ihr die Größe des Drehwinkels bestimmt?

Findet eine Regel wie ihr die Größe des Drehwinkels schnell berechnen könnt.

Übersetzt die Anweisungen dann in ein Programm am Rover.

Bringt einen Stift am Rover an und lasst ihn die Fahrt aufzeichnen. Hattet ihr Recht?

Mögliche Lösung:

*achteck.py 1/11

```
# Rover Coding
#================================
import ti_rover as rv
from math import *
import ti_plotlib as plt
from ti_system import *
from time import *
#================================
for i in range(8):
    rv.forward(2)
    rv.left(45)
```

Eine Formel für die Ermittlung des Drehwinkelmaßes lautet:

$$\frac{360°}{\text{Anzahl der Ecken}}$$

Kommentar für Lehrkräfte:

- Für jede Schüler:innengruppe ein Plakat am Boden festkleben. Stifte für die Aufzeichnung des Weges zur Verfügung stellen.

- Zur Feststellung: Die Fahrt des Rovers ist mit einer gewissen Ungenauigkeit behaftet, sodass im Regelfall am Ende nicht genau der Ausgangspunkt getroffen wird. Das Sechseck (Vieleck) ist meist nicht „geschlossen“. Die Lernenden bemerken das und korrigieren häufig das Ergebnis am Plakat mit einem Stift.

Zick-Zack-Muster (Lehrkräfte)

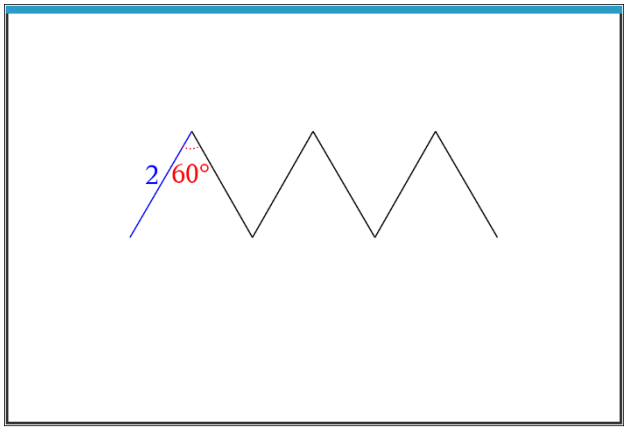
Arbeitsblatt 3

Aufgabenstellung:

Lasst den Rover das abgebildete „Zick-Zack-Muster“ fahren.

Die Länge einer einzelnen Strecke beträgt 2 Einheiten. Der von den Zacken eingeschlossene Winkel beträgt 60°.

Bringt einen Stift am Rover an und lasst ihn die Fahrt aufzeichnen.



Mögliche Lösung:

zickzack.py 1/14

```
# Rover Coding
#================================
import ti_rover as rv
from math import *
import ti_plotlib as plt
from ti_system import *
from time import *
#================================
for i in range(3):
    rv.forward(2)
    rv.right(120)
    rv.forward(2)
    rv.left(120)
```

Kommentar für Lehrkräfte:

- Für jede Schüler:innengruppe ein Plakat am Boden festkleben. Stifte für die Aufzeichnung des Weges zur Verfügung stellen.

Geschwindigkeit (Lehrkräfte)

Arbeitsblatt 4

Aufgabenstellung:

Wie schnell ist der Rover? Ermittelt die Geschwindigkeit des Rovers, indem ihr Abstandsmessungen mit dem Ultraschallsensor des Rovers und einem Hindernis (Schachtel) durchführt.

Folgende Befehle können euch dabei helfen:

rv.forward_time(time) **Tastenkombination: menu – 9 – 2 – 5 – 1**

Dieser Befehl lässt den Rover die eingegebene Anzahl an Sekunden vorwärtsfahren.

rv.wait_until_done() **Tastenkombination: menu – 9 – 7 – 5**

Dieser Befehl hält das Programm an, bis der Rover seine aktuelle Bewegungen abgeschlossen hat.

Ein Beispiel:

```
rv.forward_time(2)
rv.wait_until_done()
a=rv.ranger_measurement()
```

Dieser Code lässt den Rover zwei Sekunden lang vorwärts fahren **(1. Zeile).**
Das Programm wartet bis diese Bewegung abgeschlossen ist **(2. Zeile).**
Erst dann misst der Rover den Abstand mit dem Abstandssensor und speichert den Messwert unter der Variable *a* ab **(3. Zeile).**

Ohne dem Befehl **rv.wait_until_done()** in der 2. Zeile würde der Rover die Abstandsmessung bereits während der Fahrt durchführen und nicht wie gewünscht ganz am Ende der Fahrt.

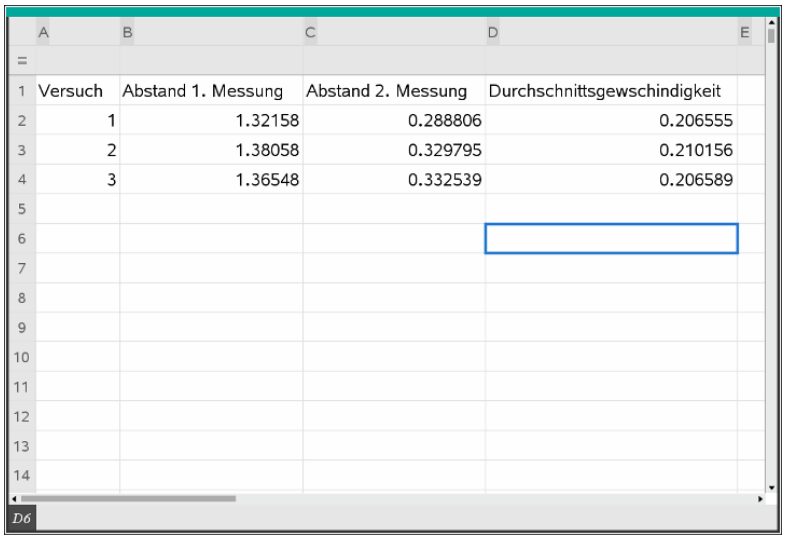
Lösungshinweis:

Das Programm lässt den Rover zwei Abstandsmessungen durchführen, eine gleich zu Beginn und eine nach einer Vorwärtsbewegung, die fünf Sekunden dauert. Angefügte Print-Befehle zeigen die Messergebnisse an.

*abstand.py

```
# Rover Coding
#================================
import ti_rover as rv
from math import *
import ti_plotlib as plt
from ti_system import *
from time import *
#================================
a=rv.ranger_measurement()
print(a)
rv.forward_time(5)
rv.wait_until_done()
a=rv.ranger_measurement()
print(a)
```

Wird das Programm mehrfach ausgeführt, so können damit beispielhaft im Rahmen von drei Versuchen folgende Daten erhoben werden:



Versuch	Abstand 1. Messung	Abstand 2. Messung	Durchschnittsgewschindigkeit
1	1.32158	0.288806	0.206555
2	1.38058	0.329795	0.210156
3	1.36548	0.332539	0.206589

Die Differenz der Abstände von der 1. zur 2. Messung gibt die zurückgelegte Strecke des Rovers in Meter an und das für eine Fahrt mit einer Dauer von fünf Sekunden (siehe Programm oben).

Hinweis: Das Komma ist hier ein Punkt.

Die Geschwindigkeit, also die zurückgelegte Strecke in Meter pro Sekunde ergibt sich aus

$$\frac{\text{Abstand 1.Messung-Abstand 2.Messung}}{\text{Dauer der Fahrt}}$$

Mit den Daten für den 1. Versuch (Zeile 2 in der Tabelle) ergibt das

$$\frac{(1{,}322-0{,}289)}{5} \approx 0{,}2$$

Die Geschwindigkeit des Rovers beträgt somit ca. 0,2 Meter pro Sekunde.

Kommentar für Lehrkräfte:

- Den Schüler:innen eine Schachtel zur Verfügung stellen.

- Ohne Angabe einer Geschwindigkeit beträgt die Standardeinstellung hierfür beim Rover 0,2 m/s.

- Die Messungen funktionieren nicht immer in jedem Fall, sodass es sinnvoll ist, mehrere Versuche durchzuführen, um etwaige Ausreißer erkennen zu können. Eine große Fahrtdauer macht es notwendig, den Rover recht weit von der Schachtel entfernt starten zu lassen, sodass auch hier größere Messungenauigkeiten oder ausbleibende Messungen (weil die Schachtel bei der 2. Messung nicht mehr „getroffen" wird) zum Tragen kommen können. Es ist empfehlenswert, eine Schachtel (anstatt einer Wand beispielsweise) für die Messungen zu verwenden, da diese bei der Wahl eines zu kleinen Abstands nicht zur Beschädigung des Rovers führt, sondern vom Rover einfach weggeschoben wird.

- Streng genommen handelt es sich bei der hier angesprochenen Geschwindigkeit des Rovers um eine Durchschnittsgeschwindigkeit, da Beschleunigungsvorgänge, insbesondere beim Anfahren und Stehenbleiben, zwangsläufig auftreten. Diese Tatsache kann als Ausgangspunkt für Reflexionen mit den Schüler:innen genutzt werden.

Suche (Lehrkräfte)
Arbeitsblatt 5

Aufgabenstellung:

Lasst den Rover von einem Punkt ausgehend den Raum nach einem Objekt (Schachtel) in seiner Umgebung absuchen, das weniger als 0.5 Meter entfernt ist. Der Rover soll dann auf das gefundene Objekt zufahren.

Folgender Befehl könnte euch dabei helfen:

rv.wait_until_done() **Tastenkombination: menu – 9 – 7 – 5**

Dieser Befehl hält das Programm an, bis der Rover seine aktuelle Bewegungen abgeschlossen hat.

Ein Beispiel:

```
rv.left(20)
rv.wait_until_done()
a=rv.ranger_measurement()
```

Dieses Code lässt den Rover eine 20°-Drehung nach links ausführen **(1. Zeile).**

Das Programm wartet bis diese Bewegung abgeschlossen ist **(2. Zeile)**.

Erst dann misst der Rover den Abstand mit dem Abstandssensor und speichert den Messwert unter der Variable a ab **(3. Zeile)**.

Hinweis:
Ohne dem Befehl **rv.wait_until_done()** in der 2. Zeile würde der Rover die Abstandsmessung bereits während der Drehung durchführen und nicht erst am Ende.

Mögliche Lösung:

*suche.py 14/15

```
# Rover Coding
#================================
import ti_rover as rv
from math import *
import ti_plotlib as plt
from ti_system import *
from time import *
#================================
a=rv.ranger_measurement()
while a>0.5:
    rv.left(10)
    rv.wait_until_done()
    a=rv.ranger_measurement()
rv.forward(2)
```

Kommentar für Lehrkräfte:
- Den Schüler:innen eine Schachtel zur Verfügung stellen.

Maximaler Abstand (Lehrkräfte)
Arbeitsblatt 6

Aufgabenstellung:

Lasst den Rover auf ein Objekt (Schachtel) zufahren und während der Fahrt den Abstand zum Objekt laufend messen, und zwar solange der Abstand zum Objekt größer als **0.5** Meter ist.

Wird der Abstand erreicht oder unterschritten, dann soll der Rover stehen bleiben

Mögliche Lösung:

*maxabstand.py 1/13

```
# Rover Coding
#================================
import ti_rover as rv
from math import *
import ti_plotlib as plt
from ti_system import *
from time import *
#================================
rv.forward(100)
a=rv.ranger_measurement()
while a>0.5:
    a=rv.ranger_measurement()
rv.stop()
```

Kommentar für Lehrkräfte:

- Den Schüler:innen eine Schachtel zur Verfügung stellen.

Konstanter Abstand (Lehrkräfte)
Arbeitsblatt 7

Aufgabenstellung:

Schreibe ein Programm zur Lösung der folgenden Problemstellung:

Ein Objekt (Schachtel) wird auf den Rover zu bzw. vom Rover wegbewegt. Der Rover soll dabei durch Vor- und Rückwärtsfahren einen Abstand *a* mit 0.45<*a*<0.55 (Angaben in Meter) zum Objekt (Schachtel) einhalten, sprich dem Objekt folgen, wenn der Abstand zu groß wird und zurückweichen, wenn der Abstand zu klein wird.

Folgende Befehle können euch dabei helfen

while get_key()!=”esc”: Tastenkombination: **menu – 9 – 7 – 4**

Solange nicht die „esc“-Taste am Rover gedrückt wird, führt das Programm fortlaufend die Befehle aus, die im zugehörigen Schleifenkörper stehen.

Ein Beispiel:

```
while get_key() != "esc":
    a=rv.ranger_measurement()
    print(a)
```

Solange nicht die „esc“-Taste am Rover gedrückt wird, misst der Rover fortlaufend den Abstand und gibt das Messergebnis aus.

Eine **Verzweigung mit zwei Bedingungen** kann ebenfalls hilfreich sein: **menu – 4 – 2 – 3**

Ein Beispiel:

```
a=rv.ranger_measurement()
if a>1:
    rv.forward(2)
elif a<0.5:
    rv.backward(2)
else:
    rv.stop()
```

Mit dieser Verzweigung wird geprüft, ob der gemessene Abstand **(1. Zeile)** größer als 1 Meter ist **(2. Zeile)**.
Trifft dies zu, dann fährt der Rover zwei Einheiten vorwärts **(3. Zeile)**.
Zudem wird auch geprüft, ob der Abstand kleiner 0.5 Meter ist **(4. Zeile)**. Ist das der Fall, dann fährt der Rover zwei Einheiten rückwärts **(5. Zeile)**.
Treffen beide Bedingungen, also a>1 und a<0.5 nicht zu **(6. Zeile)**, dann bleibt der Rover stehen **(7. Zeile)**.

Mögliche Lösung:

*konabstand.py 1/18

```
# Rover Coding
#================================
import ti_rover as rv
from math import *
import ti_plotlib as plt
from ti_system import *
from time import *
#================================
while get_key() != "esc":
    a=rv.ranger_measurement()
    if a>0.55:
        rv.forward(10)
    elif a<0.45:
        rv.backward(10)
    else:
        rv.stop()
```

Kommentar für Lehrkräfte:

- Den Schüler:innen eine Schachtel zur Verfügung stellen.