Drohnen im MINT-Unterricht in der TI-Nspire™ CAS Lernumgebung

Workshop zu physikalisch-mathematischen Grundlagen

Hans-Martin Hilbig

Ansprechpartner:
Dirk Ritschel
Policy, Teacher Training and Content Austria, Germany, Switzerland

Kontakt:
Texas Instruments EMEA Sales GmbH
Haggertystr. 1
D-85356 Freising
Mobile: +49 163/8062897
Email: d-ritschel@ti.com

© Texas Instruments EMEA Sales GmbH 2026

1. EINFÜHRUNG **3**

1.1 Vorbemerkung **3**

1.2 Sicherheits-Checkliste **4**

1.3 Schnell-Einstieg (optional) **9**

2. MODUL 1 – EINFACHE FLUGMISSIONEN **14**

2.1 Kommando-basierte Flüge **14**

2.2 Wegpunkt-basierte Flüge **16**

2.3 Bewegungsvektor-basierte Flüge **21**

3. MODUL 2 – MESSWERTERFASSUNG MIT DER CODRONE **25**

3.1 Höhenmessung oder Luftdruckmessung? **25**

3.2 Untersuchung der Restflugdauer der CoDrone in Abhängigkeit des Akkuladezustands **29**

3.3 Interaktive Objekterkennung mit der Drohne **34**

4. MODUL 3 – KOMPLEXE FLUGMANÖVER **38**

4.1 Physikalische und mathematische Grundlagen **38**

4.2 360° Selfie **40**

5. INFORMATIONEN FÜR LEHRKRÄFTE UND WORKSHOPLEITER **44**

5.1 Allgemeines **44**

5.2 Überblick über die Workshop-Übungen und Zeitbedarf **45**

5.3 Kompatibilität zwischen TI-Nspire™ und Robolink CoDrone EDU Plattform **46**

6. LITERATUR **47**

© Texas Instruments EMEA Sales GmbH 2026

1. Einführung

1.1 Vorbemerkung

Drohnen sind aus den täglichen Nachrichten kaum noch wegzudenken – meist im Zusammenhang mit aktuellen Konflikten und ihren zerstörerischen Folgen. Was dabei oft zu kurz kommt: Wie funktionieren diese Systeme eigentlich? Und welche Rolle spielen Drohnen jenseits militärischer Einsätze in unserem Alltag? Auf welchen physikalischen Gesetzmäßigkeiten beruhen die Flugmanöver einer Drohne?

Dieser Workshop setzt genau hier an. Statt spektakulärer Bilder steht das praktische Lernen und das mathematisch-naturwissenschaftliche Verständnis im Mittelpunkt: Wie fliegt ein Multikopter? Wie kann man eine Drohne per Programm-Code steuern?

Der Workshop basiert auf drei Lern-Tools: Die **TI-Nspire™** CX-Lernumgebung bildet die digitale Basis für die Programmierung der Drohne mit der Programmiersprache Python und die Auswertung der von der Drohne erzeugten Messdaten. Als Drohne für den indoor-Bereich wird die **Robolink CoDrone EDU** mit ihrem Flug-Controller verwendet. Zur Kommunikation zwischen der TI-Nspire™ Software und der CoDrone Drohne eignet sich ein **TI-Nspire™ CX II Taschenrechner** ebenso wie die **TI-Nspire™ CX-Lernumgebung** auf einem Windows-PC oder einem Apple-Mac. Natürlich kann auch die *TI-Nspire™ CAS*-Technologie genutzt werden.

Die für diesen Workshop nötigen Grundlagen in der Handhabung der CoDrone in der TI-Nspire™ CX-Lernumgebung entnehmen Sie bitte dem TI-Nspire™ CX II CoDrone EDU Einführungshandbuch[1].

Aufgrund ihres geringen Fluggewichts von nur 58 g und den in Propeller-Käfigen geschützten Propeller eignet sich die CoDrone gut zum Einsatz in geschlossenen Räumen für praktische Übungen und Workshops. Im Gegensatz zu anderen Indoor-Drohnen besitzt die
CoDrone keine Kamera, daher ist ihr Einsatz auch unbedenklich hinsichtlich der Datenschutz-Grundverordnung (DSGVO).

[1] TI-Nspire™ CX II – CoDrone EDU Einführungshandbuch:
https://education.ti.com/de/product-resources/codrone-edu

© Texas Instruments EMEA Sales GmbH 2026

1.2 Sicherheits-Checkliste

Der Betrieb von Drohnen wird in Deutschland durch die seit Anfang 2024 in Kraft getretene EU-Drohnenverordnung geregelt. Das Luftfahrtbundesamt (LBA) ist in Deutschland die zuständige nationale Luftfahrtbehörde für die Umsetzung und Anwendung der EU-Drohnenverordnung[2]. Seine Funktion ist nicht die Gesetzgebung, sondern die administrative Umsetzung, Überwachung und Vollziehung der geltenden EU-Regeln.

Die EU-Drohnenverordnung sieht verschiedene Kategorien für den Einsatz von Drohnen vor. Diese Kategorien unterscheiden sich nach drei grundlegenden Kriterien:

1. Gefährdungspotential (Risikoklasse, abhängig von Ausstattung, Gewicht und Einsatz)
2. Startgewicht der Drohne (Maximum Take-Off Mass, MTOM)
3. Ausrüstung der Drohne mit einem Aufnahmegerät (Kamera, Mikrofon)

Drohnen mit einem Startgewicht unter 250g und Fehlen eines Aufnahmegeräts, wie bei der CoDrone der Fall, sind registrierungsfrei und fallen nicht unter die EU-Drohnenverordnung. Trotzdem sollte der Schulleiter oder die verantwortliche Person für den Workshop-Raum vorab informiert werden.

Ebenso ist es eine gute Praxis zur Vorbereitung der Schüler, vor Beginn des Workshops eine Sicherheits-Checkliste abzuarbeiten. Nachfolgend ein Beispiel für eine Sicherheits-Checkliste für den CoDrone Workshop.

[2] Luftfahrtbundesamt (LBA) Die neue Drohnen-Verordnung - Ein Überblick über die wichtigsten Regeln (BMDV)
https://www.lba.de/SharedDocs/Downloads/DE/B/B5_UAS/Drohnenflyer_BMVI.html

© Texas Instruments EMEA Sales GmbH 2026

Bereich	Checkpunkt	Beschreibung	Notizen/abgehakt
Raum	Informationspflicht	Schulleiter bzw. verantwortliche Person über die Veranstaltung des CoDrone-Workshops informiert?	
Pilot/Crew	Persönliche Ausrüstung & Hilfsmittel vorhanden?	Brille, Schreibzeug für Notizen, Unterlagen, usw.	
	Rollen klar definiert?	z. B. Programmierer, Flugbeobachter, Bediener	
	Physische und psychische Verfassung?	Voraussetzung für die jeweilige Flugmission	
	Alle Systeme vorhanden und bereit?	TI-Nspire™ Software, PC, CoDrone, Controller, Batterien, Ladegerät, Spezial-Verbindungskabel für TI-Nspire™ CX II Handheld und Codrone Controller?	
Drohne	Sichtprüfung	Drohnen-Gehäuse und Propellerkäfige unbeschädigt?	
		Propeller vorhanden und unbeschädigt?	
		Alle vier Rotoren lassen sich frei drehen?	
		Akku vorhanden und unbeschädigt?	
Umgebung der Flugmission	Keine Hindernisse im geplanten Flugbereich?	z. B. empfindliche Gegenstände, leichte Gegenstände, die vom Luftstrom der Rotoren verwirbelt werden könnten, unbeteiligte Personen, Gegenstände, mit denen die Drohne kollidieren könnte?	
	Raum-Regeln	Sind die für den benutzten Raum gültigen Regeln bekannt und	

© Texas Instruments EMEA Sales GmbH 2026

		werden sie befolgt? Gibt es „no-fly“-Zonen?	
Energieversorgung	Drohnen-Akku, Controller-Akku, TI-Nspire™ CX II Handheld-Akku	Alle Akkus ausreichend geladen? Ladegeräte für Drohnen-Akku und TI-Nspire™ CX II-Akku verfügbar, Ersatzbatterien für CoDrone Controller verfügbar?	

© Texas Instruments EMEA Sales GmbH 2026

Bereich	Checkpunkt	Beschreibung	Notizen/abgehakt
Überwachung der Flugmission	Welche kritischen Sensor-Daten sollten vom Python Programm überwacht werden?	z. B. Batterie-Ladezustand während des Flugs, Flughöhe der Drohne, Position der Drohne	
	Sicherheitsfunktionen im Python-Programm implementiert?	z. B. Abbruch der Flugmission und Landung der Drohne bei Auftreten kritischer Sensor-Daten?	
	Flugmissionsüberwachungsperson bestimmt?	Ein Schüler konzentriert sich auf die visuelle Überwachung der Flugmission	
	Überwachungsperson ist mit dem Not-Stopp-Prozess vertraut?	L1-Taste am CoDrone Controller gedrückt halten und linken Steuerknüppel nach unten ziehen.	
Checks nach dem Ende der Flugmission	Landung	Drohne sicher am geplanten Zielpunkt gelandet?	
	Ausschalten der Geräte	z. B. Akku von der CoDrone trennen, um bis zur nächsten Mission Energie zu sparen.	
	Flugmissions-Daten sichern	Wichtige Flugdaten gesichert, z. B. durch Screenshots der Python-Shell,	

© Texas Instruments EMEA Sales GmbH 2026

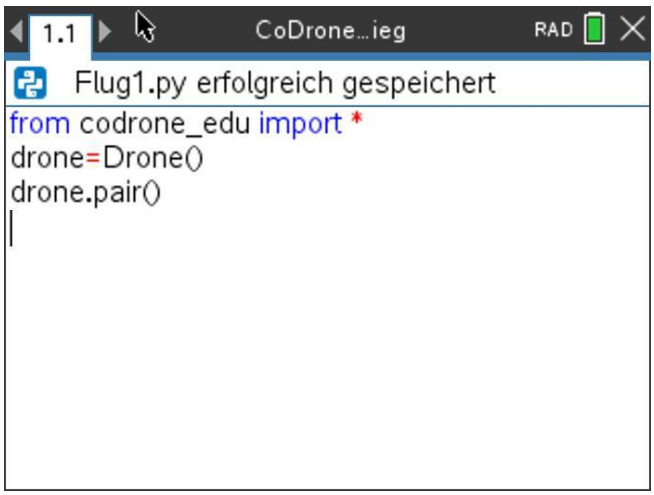
		Speichern des TI-Nspire™ Dokuments? Speichern der aufgenommen Messdaten in Nspire-Variablen?	

1.3 Schnell-Einstieg (optional)

Für Workshop-Neueinsteiger, die noch nie mit dem TI-Nspire™ CoDrone System gearbeitet haben, empfiehlt sich, das TI-Nspire™ CX II – CoDrone EDU Einführungshandbuch zu lesen. Dort werden ausführlich die notwendigen Installationsschritte beschrieben, um das Robolink CoDrone System mit dem TI-Nspire™ System zu verbinden. Im Abschnitt „Erste Schritte“ wird ein erstes Python-Programm für die CoDrone angeboten, das sich gut für den ersten Praxis-Test von Hard- und Software eignet[3].

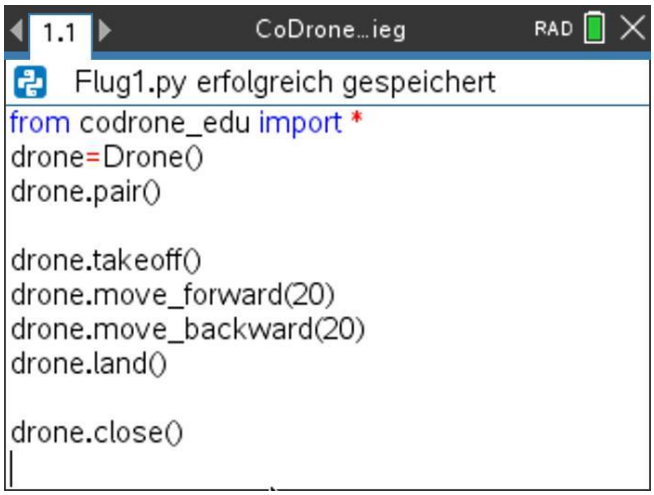
Zur weiteren Vertiefung und für einen Überblick über die Möglichkeiten des TI-Nspire™ CoDrone Systems kann auch der nachfolgende Schnell-Einstieg hilfreich sein:

Schritt 1: Mit der Drohne 20 cm vorwärts und rückwärts fliegen

Schritte	Python Code-Fenster
1. Erzeugen Sie ein neues Python Projekt und vergeben Sie einen sinnvollen Namen, zum Beispiel „Flug1“: Über das Menu des TI-Nspire™ CX II wählen Sie folgende Kommandozeile aus. **Menu →** **Weitere Module →** **Robolink Codrone EDU →** **from codrone_edu import *** Dieser Befehl importiert das codrone_edu Modul, erzeugt ein Python ‘drone’-Objekt und ergänzt Code, um die Drohne mit dem TI-Nspire™ CX II Handheld zu koppeln.	

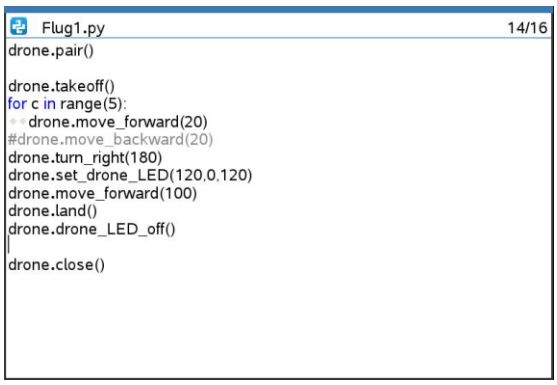
[3] TI-Nspire™ CX II – CoDrone EDU Einführungshandbuch https://education.ti.com/de/product-resources/codrone-edu

2. Lassen Sie die Drohne abheben, jeweils 20 cm vorwärts und rückwärts fliegen und wieder landen: **Menu →** **Weitere Module →** **Robolink Codrone EDU →** **Flight (start/stop)** **Menu →** **Weitere Module →** **Robolink Codrone EDU →** **Flight (movements)**	

Schritte	Python Code-Fenster
3. Beenden Sie die Kopplung der Drohne mit dem TI-Nspire™ CX II: **Menu →** **Weitere Module →** **Robolink Codrone EDU →** **Flight (start/stop) →** **drone close** Probieren Sie ihr erstes Programm zum Steuern der CoDrone aus! Startet die Drohne, fliegt diese 20 cm vorwärts, dann 20 cm rückwärts und landet wieder? Sehr gut, dann erweitern Sie ihr Programm im nächsten Schritt.	

© Texas Instruments EMEA Sales GmbH 2026

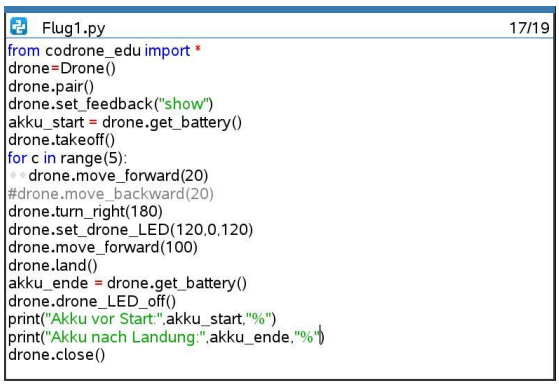
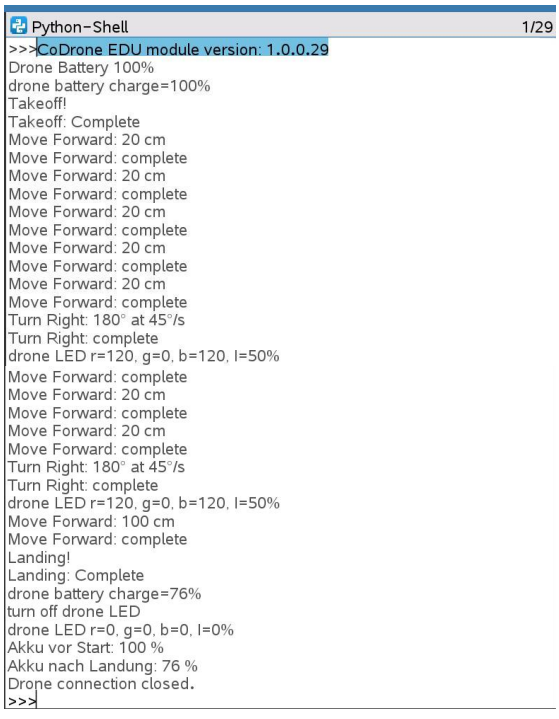
Schritt 2: Mit der Drohne vorwärts fliegen, umdrehen und zurückfliegen, landen und Beleuchtung der Drohne ausschalten

Schritte	Python Code-Fenster
1. Eine For-Schleife einbauen. Fügen Sie vor der drone.move_forward(20) Zeile eine For-Schleife ein: **Menu → Eingebaute → Steuerung →** **for index in range(size):** Drücken Sie <Enter> und beobachten Sie, wie die nächste Zeile automatisch eingerückt wurde. Drücken Sie die <Entf>-Taste, um das drone.move_forward() Kommando in die For-Schleife zu bewegen. 2. Das Rückwärtsflug-Kommando „aus-kommentieren“: Mit dem Cursor zum Anfang der drone.move_backward() Zeile navigieren und ein „#“ einfügen. Das ‚#‘-Zeichen lässt das Python-Programm diese Zeile überspringen. 3. Die Drohne um 180° drehen lassen. Fügen Sie nach der For-Schleife das Kommando zum Drehen der Drohne ein. **Robolink Codrone EDU → Flight (movement)** 4. Die LED der Drohne auf eine andere Farbe umschalten. Diese Funktion benötigt drei Parameter, die die Helligkeit der einzelnen Farben (Rot, Grün, Blau) der RGB-LED in 256 Helligkeitsstufen (0-255) definieren. Hier im Beispiel ist dies „Violett“ in mittlerer Helligkeit. Fügen Sie das entsprechende Kommando ein. **Robolink Codrone EDU → LED and Buzzer** 5. Das drone.move_backward() Kommando durch ein drone.move_forward() Kommando ersetzen und die Drohne zurück zum Startplatz fliegen lassen.	

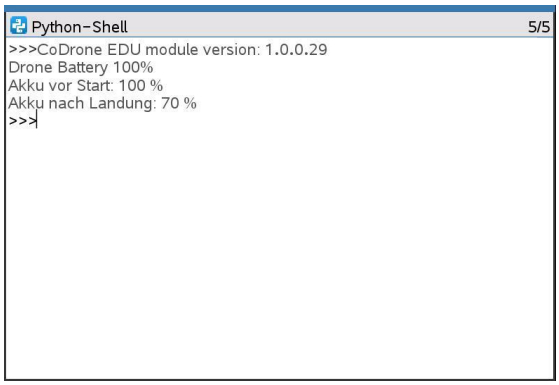
Schritte	Python Code-Fenster
6. Zuletzt schalten Sie die Drohnen-LED nach der Landung aus. Probieren Sie ihren neuen Code aus und beobachten Sie, ob die Drohne den Kommandos folgt.	

Schritt 3: Sensor-Messdaten aufnehmen und anzeigen

© Texas Instruments EMEA Sales GmbH 2026

Schritte	Python Code-Fenster
1. Ergänzen Sie nach der Kopplung der Drohne eine Zeile zur Steuerung des Ausgabeprotokolls in die Shell: **Menu → Robolink Codrone EDU →** **Commands → set_feedback(Show/Hide)** Im Normalfall ist der Protokoll-Modus eingeschaltet ("show"). 2. Ergänzen Sie zwei Sensor-Messungen, um den Akku-Ladezustand zu Beginn und am Ende der Mission zu messen: **Menu → Robolink Codrone EDU →** **Sensors → variable = get_battery()** 3. Ergänzen Sie zwei Befehle zur Ausgabe des gemessenen Akku-Ladezustands, nachdem die Drohne gelandet ist, aber vor dem drone.close() Befehl. Probieren Sie ihr neues Programm aus und schauen sich danach das Ausgabeprotokoll in der Shell an.	
4. Sie sollten nebenstehendes Protokoll in der Shell sehen. Machen Sie einen Screenshot der Shell. Wenn die Protokoll-Funktion eingeschaltet ist, wird jeder Steuerungsbefehl der Drohne im Shell-Protokoll angezeigt. Dies ist hilfreich, um neuen Code auszutesten und man spart sich print() Befehle. Andererseits kostet die Protokoll-Ausgabe Zeit und verlangsamt die Ausführung des Python-Programms. Daher ist es manchmal ratsam, die Protokoll-Funktion mit dem Befehl drone.set_feedback("hide") auszuschalten.	

© Texas Instruments EMEA Sales GmbH 2026

5. Schalten Sie die Feedback-Funktion aus. **Menu → Robolink Codrone EDU →** **Commands → set_feedback(“hide”)** Probieren Sie ihren neuen Code aus und vergleichen Sie den Shell-Inhalt mit dem Screenshot der vorherigen Übung.	

2. Modul 1 – Einfache Flugmissionen

2.1 Kommando-basierte Flüge

i. Hintergrund und Aufgabe

Zu den kommando-basierten Flugmanövern gehören alle Bewegungen in Richtung der Längsachse (Vorwärts, Rückwärts), der Querachse (Links, Rechts) oder der Hochachse (Drehen, Steigen, Fallen) der Drohne.

Schreiben Sie ein Python-Programm, um die CoDrone zu starten, ein Quadrat mit 50cm Kantenlänge zu fliegen und wieder am Startpunkt zu landen. Während des Fluges soll die Nase der Drohne immer in Flugrichtung zeigen.

ii. Lernziel

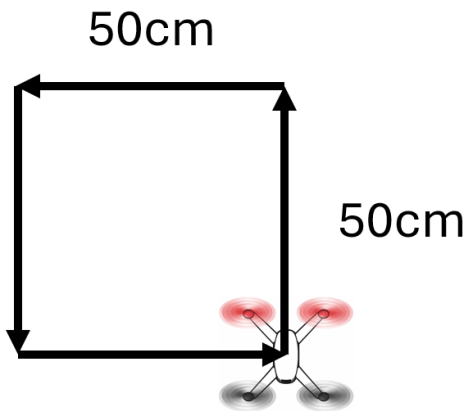
Anwendung von kommando-basierten Flugmanövern, um geometrische Flugbahnen zu fliegen. Schleifenprogrammierung in Python.

iii. Zielgruppe

Schüler und Schülerinnen ab der 5. Jahrgangsstufe mit geringen Programmier-Kenntnissen.

iv. Anleitung und Code

Wählen Sie einen Startplatz der Drohne, von dem gefahrlos und ohne Hindernisse ein Quadrat mit 50 cm Kantenlänge abgeflogen werden kann.

Schritte	Python Code-Fenster
1. Fertigen Sie eine Skizze der gesamten Flugmission an. Teilen Sie die Flugmission in sinnvolle Teilschritte, die im Python Code per Schleife programmiert werden können. Notieren Sie sich für jeden Teilschritt die Flugbewegungen, die die Drohne durchführen muss.	

© Texas Instruments EMEA Sales GmbH 2026

2. Erzeugen Sie ein neues Python Fenster und vergeben Sie einen sinnvollen Namen, zum Beispiel „Quadrat“. Speichern Sie das gesamte TI-Nspire™ Projekt unter dem Namen „Modul1“. Speichern Sie Ihr Projekt regelmäßig, um Datenverlust zu vermeiden.

Über das Menu des TI-Nspire™ CX II wählen Sie folgende Kommandozeile aus.

Menu →

Weitere Module →

Robolink Codrone EDU →

from codrone_edu import *

Dieser Befehl importiert das codrone_edu Modul, erzeugt ein Python ‘drone’-Objekt und ergänzt Code, um die Drohne mit dem TI-Nspire™ CX II Handheld zu koppeln.

3. Ergänzen Sie die Flugmanöver, um die Drohne abheben zu lassen und 50 cm nach vorne zu fliegen.

Menu →

Weitere Module →

Robolink Codrone EDU →

Flight (start/stop)

Menu →

Weitere Module →

Robolink Codrone EDU →

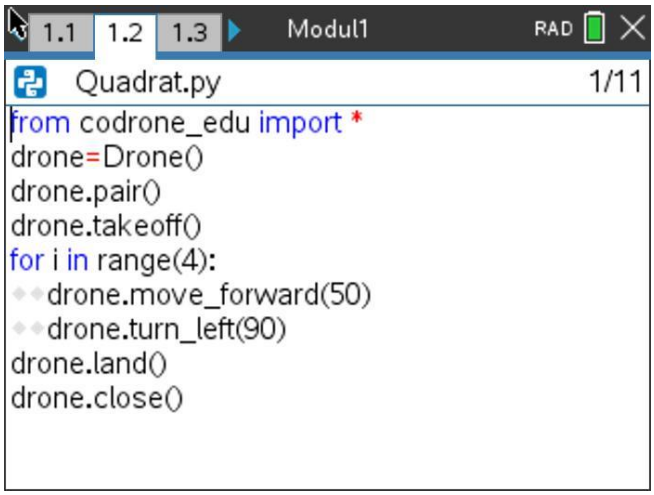
Flight (movements)

4. Fügen Sie ein „For“-Schleife ein, um alle vier Seiten des Quadrats abzufliegen. Achten Sie dabei darauf, dass die Nase der Drohne immer in Flugrichtung zeigt.

Menu → Eingebaute → Steuerung →

for index in

range(size):



v. Reflektion, Gelerntes

Diskutieren Sie in der Gruppe Ihre Erfahrungen bei dieser ersten Übung. Hat alles auf Anhieb funktioniert? Wo gab es Schwierigkeiten, wobei wurde Hilfe benötigt? Sind irgendwelche unvorhergesehene Dinge passiert, wenn ja, was waren die Ursachen? Wie kann man diese unvorhergesehenen Situationen bei der nächsten Übung vermeiden?

Nach dieser Übung sollten Sie schon deutlich vertrauter mit der Handhabung der CoDrone und der TI-Nspire™ Lernumgebung geworden sein. Sie sollten in der Lage sein, auf Basis dieser Übung auch andere geometrischen Figuren als Flugmission zu programmieren.

Anregungen und Erweiterungen

Ändern Sie den Startpunkt der Drohne und die Flugrichtung zum Abfliegen des Quadrats, d. h., wenn die Drohne bisher das Quadrat gegen den Uhrzeigersinn geflogen ist, lassen Sie die Drohne im Uhrzeigersinn fliegen.

Verlegen Sie Startpunkt und Zielpunkt der Drohne in die Mitte einer Kante des Quadrats.

Stellen Sie sich vor, dass die Drohne eine Inspektionsdrohne wäre, die den Innenraum des Quadrats von allen vier Seiten abfotografieren soll. Ändern Sie ihr Programm, dass die Nase der Drohne beim Abfliegen aller vier Seiten immer ins Innere des Quadrats zeigt.

Probieren Sie auch andere Figuren Ihrer Flugmission aus, z. B. Dreieck, Fünfeck, usw. Entwickeln Sie Python Code, der nach Vorgabe der Anzahl der Ecken automatisch die richtigen Winkel und die Kantenlänge der zu fliegenden Figur berechnet.

2.2 Wegpunkt-basierte Flüge

i. Hintergrund und Aufgabe

Wegpunkt-basierte oder auch POI (Point Of Interest)-Flugmanöver sind die am häufigsten genutzten Manöver beim Einsatz von automatischen Missionen von Drohnen im industriellen Bereich. Die entsprechende Funktion bewegt die Drohne zu einem genau definierten Punkt im drei-dimensionalen Koordinatensystem. Die Flugbahn ist der Vektor von der aktuellen Position der Drohne zum (x, y, z)-Zielpunkt. Eventuell vorhandene Hindernisse auf der Vektor-Linie vom Start- zum Zielpunkt stellen das größte Risiko bei diesen Flugmanövern dar. Während kommerziell eingesetzte Drohnen sich mit einem GPS-Signal im drei-dimensionalen Koordinatensystem orientieren, nutzt die CoDrone drei ihrer eingebauten Sensoren: Die Mustererkennung des nach unten gerichteten Sensors zur Orientierung in x- und y-Richtung, den Abstandssensor nach unten und den barometrischen Sensor zur Bestimmung der Höhe in z-Richtung.

Nutzen Sie Ihr Python-Programm aus Übung (1.1) und verwenden Sie vier Wegpunkt-basierte Funktionen, um die vier Ecken des Quadrats mit 50 cm Kantenlänge anzufliegen und wieder am Startpunkt zu landen.

Im zweiten Schritt erweitern Sie Ihr Programm, um beliebig viele, in einem TI-

© Texas Instruments EMEA Sales GmbH 2026

Nspire™ -Lists & Spreadsheet-Fenster eingetragenen Wegpunkte abzufliegen.

ii. Lernziel

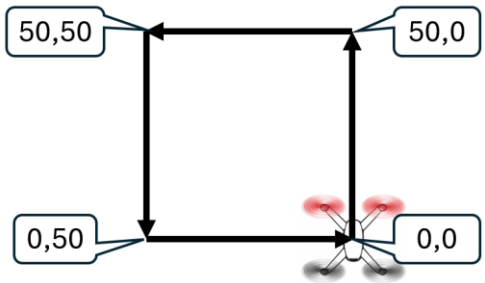
Anwendung von Wegpunkt-basierten Flugmanövern, um beliebige geometrische Flugbahnen im drei-dimensionalen Koordinatensystem zu fliegen. Schleifenprogrammierung in Python, den Python zip()-Befehl. Verwendung von TI-Nspire™ Lists & Spreadsheet für die Flugmission der CoDrone.

iii. Zielgruppe

Schüler und Schülerinnen ab der 5. Jahrgangsstufe mit geringen Programmier-Kenntnissen.

iv. Anleitung und Code

Wählen Sie einen Startplatz der Drohne, von dem gefahrlos und ohne Hindernisse beliebige Koordinaten in einem 100 x 100 x 200 cm (x, y, z) Raum angeflogen werden können.

Schritte	Python Code-Fenster
1. Nehmen Sie die Skizze aus Übung (1.1) zur Hand und tragen Sie die x- und y-Koordinaten der vier Eckpunkte des 50 x 50 cm Quadrats ein. Bestimmen Sie eine feste Höhe, in der alle Eckpunkte angeflogen werden sollen, z. B. 80 cm.	

2. Öffnen Sie ihr Python Projekt „Modul1“ und addieren Sie ein neues Python-Fenster, zum Beispiel „POI_mission1“. Gehen Sie zum bereits vorhandenen Python-Fenster „Quadrat“, klicken in den Python Code und drücken Sie <strg>a, um den gesamten Code auszuwählen. Drücken Sie <strg>c, um den Code zu kopieren. Wechseln Sie in das neue leere Python-Fenster „POI_mission1“ und drücken Sie <strg>v. Nun sollte der Code vom „Quadrat“-Fenster erscheinen. Ersetzen Sie die Flugmanöver und die Schleife durch vier Wegpunkt-Anweisungen mit den unter (1.) notierten Koordinaten. Speichern Sie ihr Projekt mit dem neuen Python-Fenster.	POI_mission1.py
	`from codrone_edu import *`
	`drone=Drone()`
	`drone.pair()`
	`drone.takeoff()`
	`drone.move_to_xyz(50,0,80)`
	`drone.move_to_xyz(50,50,80)`
	`drone.move_to_xyz(0,50,80)`
	`drone.move_to_xyz(0,0,80)`
	`drone.land()`
	`drone.close()`

3. Zeichnen Sie „Das Haus vom Nikolaus“ in der x-y-Ebene. Vergeben Sie Koordinaten für die fünf Eckpunkte des Hauses innerhalb des 100 x 100 x 150 cm Luftraums. Ermitteln Sie die nötige Anflugsequenz, um alle Elemente vom „Haus vom Nikolaus“ in einem Zug abzufliegen. Erzeugen Sie ein Lists & Spreadsheet-Fenster, benennen Sie drei Spalten des Spreadsheets mit x, y, z und tragen Sie die Wegpunkt-Koordinaten in der für die Anflugsequenz richtigen Reihenfolge ins Spreadsheet ein. Definieren Sie Variablen „x“,“y“,“z“ für die Spalten A, B, C.

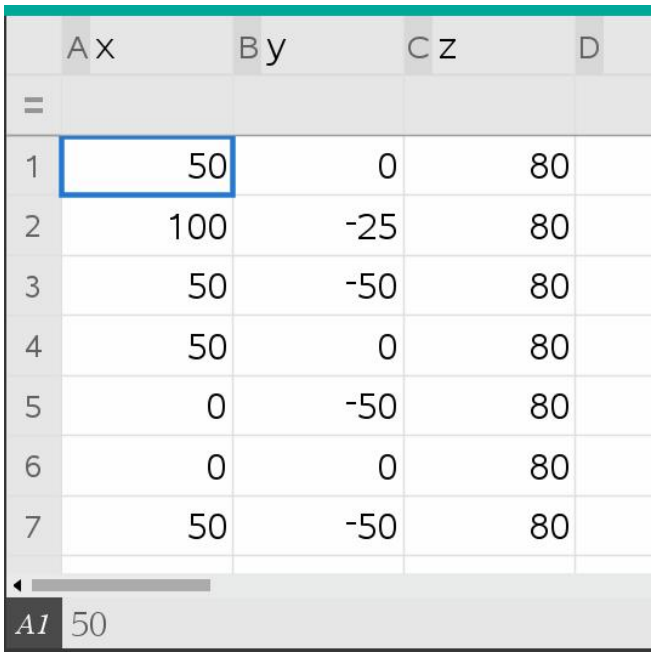
Einfügen →

Lists & Spreadsheet

In das Feld mit dem Spaltennamen klicken→

<strg><var> drücken→

Den Variablen-Namen eingeben



	A x	B y	C z	D
=				
1	50	0	80	
2	100	-25	80	
3	50	-50	80	
4	50	0	80	
5	0	-50	80	
6	0	0	80	
7	50	-50	80	

A1 50

4. Erzeugen Sie ein weiteres Python-Fenster und benennen Sie es „POI-mission2“. Kopieren Sie den Code von „POI_mission1“ in das „POI_mission2“-Fenster, um eine Basis für den Code für die „Haus vom Nikolaus“-Mission zu haben. Fügen Sie den für die Umwandlung von Nspire-Listen in Python-Listen nötigen Code ein.

Menu →

Weitere Module →

Robolink Codrone EDU →

Flight (movements)

Ersetzen Sie die vier Wegpunkt-basierten Flugmanöver durch eine Schleife, die die Wegpunktliste abarbeitet.

POI_mission2.py

```
from codrone_edu import *
drone=Drone()
drone.pair()
drone.takeoff()
x = recall_list("x")
y = recall_list("y")
z = recall_list("z")
poi_liste = list(zip(x, y, z))
for poi in poi_liste:
    print("Wegpunkt:",poi)
    drone.move_to_xyz(poi[0],poi[1],poi[2])
drone.land()
drone.close()
```

© Texas Instruments EMEA Sales GmbH 2026

v. Reflektion, Gelerntes

Diskutieren Sie in der Gruppe Ihre Erfahrungen bei dieser Übung. Wie hat sich die Drohne beim Abfliegen des Quadrats verhalten? Wohin hat die Nase der Drohne beim Anflug auf die Wegpunkte gezeigt? Hat Alles auf Anhieb funktioniert? Wo gab es Schwierigkeiten, wobei wurde Hilfe benötigt? Sind irgendwelche unvorhergesehene Dinge passiert, wenn ja, was waren die Ursachen? Wie kann man diese unvorhergesehenen Situationen bei der nächsten Übung vermeiden? Haben Sie die Funktion der in dieser Übung neu hinzugekommenen Python Befehle (Lists & Spreadsheet-Fenster, Nspire- vs. Python-Listen, Abarbeiten von Listen, den zip()-Befehl) verstanden? Wenn Sie mit einzelnen Python-Funktionen noch unsicher sind, üben Sie diese Funktionen im Shell-Fenster. Dort können Sie direkt das Resultat der Funktionen sehen.

vi. Anregungen und Erweiterungen

In den Übungen wurden die Wegpunkte in der zwei-dimensionalen (x-y)-Ebene angeflogen. Um mit dem 3-dimensionalen Koordinatensystem vertraut zu werden, ändern Sie Übung 4 und lassen Sie die Drohne das „Haus vom Nikolaus“ in der y-z-Ebene (vertikal) abfliegen.

Erweitern Sie Ihr Python-Programm für eine mathematisch unterstützte Berechnung des Höhenprofils Ihrer Wegpunkte.

Beispiel: Die Drohne steht auf dem Eckpunkt eines Quadrates, der im Koordinatenursprung liegt. Die Seitenlänge beträgt 50 cm, je eine Seite des Quadrats liegt auf der x- bzw. y-Achse.

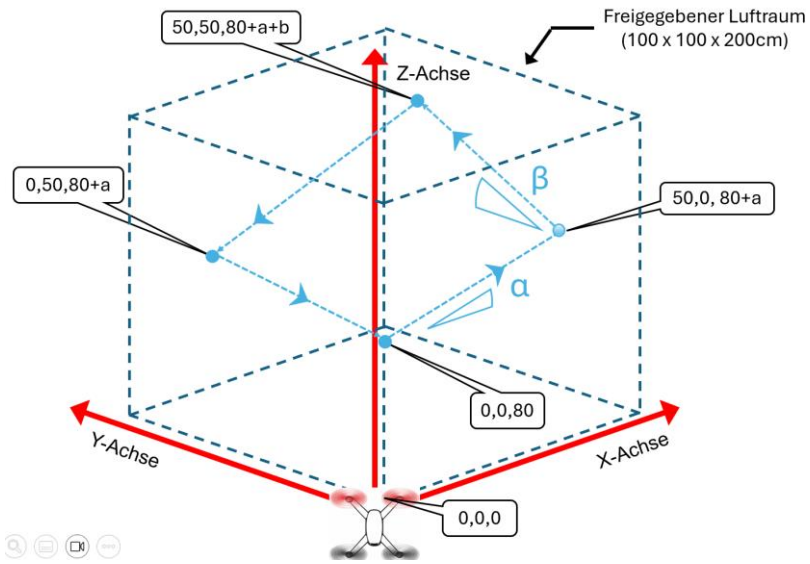
Die Drohne soll alle Seiten des Quadrats folgendermaßen überfliegen:

1. Senkrechter Aufstieg um 80 cm.
2. Fliege in positive x-Richtung mit einer Steigung von 20° bis direkt über den nächsten Eckpunkt.
3. Fliege dann mit einer Steigung von 30° in positive y-Richtung um direkt über den nächsten Eckpunkt zu gelangen.
4. Von dort fliege mit gleichmäßiger Neigung im Sinkflug über den dritten Eckpunkt des Quadrates zurück zum Punkt mit den Koordinaten (0; 0; 80) und lande die Drohne.

Erfasse jeweils die Koordinaten der Drohne, wenn sie über den Eckpunkten des Quadrates schwebt.

Probieren Sie Ihren Code zunächst in einem separaten Python-Programm aus und binden Sie ihn anschließend in Ihr Programm aus Übung 4 ein. Was sind die maximal zulässigen Steigungswinkel, bei der die maximal erlaubte Flughöhe von 200 cm im Flugraum nicht überschritten wird?

© Texas Instruments EMEA Sales GmbH 2026


50,50,80+a+b
Freigegebener Luftraum
(100 x 100 x 200cm)
Z-Achse
0,50,80+a
β
50,0, 80+a
α
0,0,80
Y-Achse
X-Achse
0,0,0

2.3 Bewegungsvektor-basierte Flüge

i. Hintergrund und Aufgabe

Die CoDrone bietet noch eine dritte Gruppe von möglichen Flugmanövern an, nennen wir sie die Bewegungsvektor-basierten Flüge. Diese Flugart simuliert die Steuerung über die Steuerknüppel der Fernsteuerung, bei der gleichzeitig Nick-, Roll- Rotations- und Schub-Parameter beeinflusst werden können. Die physikalischen Grundlagen werden im Modul 3 dieses Workshops näher erläutert. Hier in dieser Aufgabe geht es um einfache Grundübungen, deren Ergebnisse später für Modul 3 hilfreich sind. Ebenso wird in dieser Aufgabe Sensor-Daten der CoDrone aufgenommen und dargestellt, also eine gute Grundübung für das Modul 2, in dem es um das Thema „Messwerterfassung“ geht.

Bestimme experimentell mit der „.set_yaw()“ Funktion die Winkelgeschwindigkeit der Drohne bei Drehung um die Hochachse in Abhängigkeit von der eingestellten Leistung.

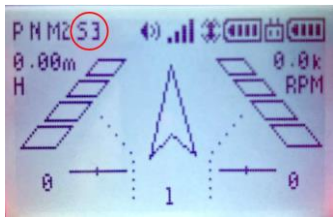
ii. Lernziel

Anwendung von Bewegungsvektor-basierten Flugmanövern, um kurvenförmige Flugbahnen zu fliegen. Messwerterfassung und Auswertung mit dem TI-Nspire™ - System.

iii. Zielgruppe

Schüler und Schülerinnen ab der 7. Jahrgangsstufe mit Erfahrung in einer Programmiersprache, vorzugsweise Python.

iv. Anleitung und Code



Sichern Sie sich einen Flugraum von 100 x 100 x 200 cm, in dem gefahrlos und ohne Hindernisse beliebige Flugmanöver geflogen werden können. **Wichtig: Bevor Sie den CoDrone Flugcontroller mit dem TI-Nspire™ System verbinden, stellen Sie bitte den Controller auf Geschwindigkeitsstufe S3, indem Sie die L1-Taste des Controllers wiederholt drücken, bis „S3“ im Display erscheint.**

Schritte	Python Code-Fenster

1. Erzeugen Sie ein neues Python Fenster in Ihrer Modul1.tns Datei und vergeben Sie einen sinnvollen Namen, zum Beispiel „Winkelgeschwindigkeit1“. Speichern Sie das gesamte TI-Nspire™ Projekt unter dem Namen „Modul1“. Speichern Sie Ihr Projekt regelmäßig, um Datenverlust zu vermeiden.
Die für Bewegungsvektor-basierte Flüge relevanten Funktionen finden Sie im Menu des TI-Nspire™ CX II Handhelds:
Menu →
Weitere Module →
Robolink Codrone EDU →
Flight (controls) →
	Winkelgeschwindigkeit1.py
	`from codrone_edu import *`
	
	`leistung = 50 # Leistung in %`
	`dauer = 1 # Ausführungsdauer in Sekunden`
	
	`drone=Drone()`
	`drone.pair()`
	`drone.takeoff()`
	`drone.set_yaw(leistung)`
	`drone.move(dauer)`
	`omega = drone.get_heading()`
	`drone.land()`
	`drone.close()`

Führen Sie den Code aus. Der Inhalt der Shell sollte so ähnlich wie rechts im Bild aussehen. Bei einer Leistung von 50 % hat sich die Drohne nach einer Sekunde um gut 90° gedreht, dies entspricht einer Winkelgeschwindigkeit von 90°/s.	Python-Shell Drone Battery 100% Takeoff! Takeoff: Complete Set yaw to 50 Move for 1s Move: Complete heading =93° Landing! Landing: Complete Drone connection closed. >>>

2. Untersuchen Sie als nächstes, in welchem Verhältnis sich die Winkelgeschwindigkeit in Abhängigkeit der Leistung ändert. Beginnend bei 10 %, steigern Sie die Leistung in 10 %-Schritten bis zu 100 % Leistung. Messen Sie nach jedem Schritt die Richtung der Drohne. Speichern Sie jedes Wertepaar aus Leistung und Richtung in Nspire-Variablen, um die Messwerte später in der TI-Nspire™ -Umgebung darstellen und analysieren zu können.
Hinweis: Bei jedem take-off-Befehl wird das Heading (die Richtung) des Winkelsensors der CoDrone zurückgesetzt. Es ist daher wichtig, nach jeder Winkeländerung kurz zu landen, um den Winkelsensor wieder auf „0“ zurückzusetzen. Erweitern Sie Ihr Programm entsprechend der Aufgabenstellung, z. B. wie abgebildet.
 	
Führen Sie das Programm aus.
	Winkelgeschwindigkeit2.py
	`from codrone_edu import *`
	`dauer = 1 # Ausführungsdauer in Sekunden`
	`drone=Drone()`
	`drone.pair()`
	`heading = []`
	`leistung = []`
	`for i in range(10):`
	`  drone.takeoff()`
	`  drone.set_yaw((i+1)*10)`
	`  drone.move(dauer)`
	`  heading.append(drone.get_heading())`
	`  leistung.append((i+1)*10)`
	`  drone.land()`
	`store_list("Heading",heading)`
	`store_list("Leistung",leistung)`
	`drone.close()`

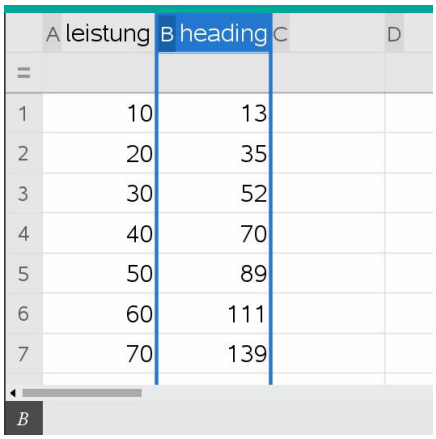
© Texas Instruments EMEA Sales GmbH 2026

Fügen Sie ein Lists & Spreadsheet-Fenster in der TI-Nspire™ -Umgebung ein. Verknüpfen Sie die Spalte A mit der Nspire-Variablen „leistung“ und die Spalte B mit der Nspire-Variablen „heading“, indem Sie die Taste <Var> drücken:

Var →

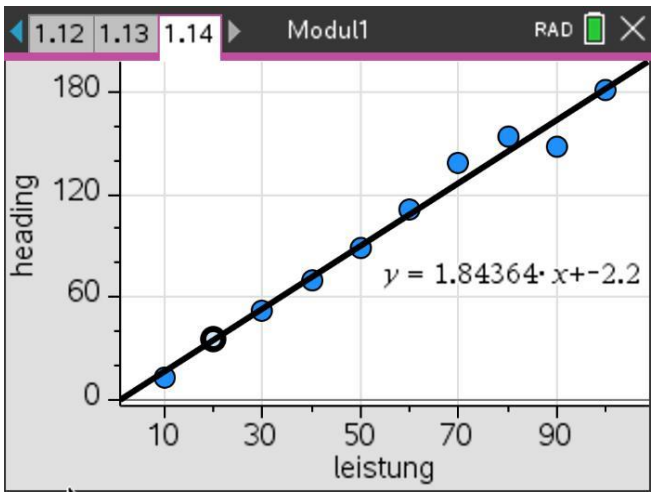
Verknüpfen mit: →

leistung



	A leistung	B heading	C	D
1	10	13		
2	20	35		
3	30	52		
4	40	70		
5	50	89		
6	60	111		
7	70	139		

Fügen Sie ein Data & Statistics-Fenster ein. Verknüpfen Sie die x-Achse mit „leistung“ und die y-Achse mit „heading“. Legen Sie eine Regressionsgerade über die angezeigten Werte.



3. Falls gewünscht, kann die Winkelgeschwindigkeit in Abhängigkeit der Leistung auch ohne Zwischenlandung gemessen werden. Hierbei sind jedoch zwei Dinge zu beachten:
a) Das Heading der Drohne wird nicht zurückgesetzt, sondern die Werte kumulieren. Um die absoluten Werte zu erhalten, muss man den vorherigen Wert vom aktuellen Wert subtrahieren.
b) Wenn das kumulative Heading 360° übersteigt, läuft der Heading-Zähler über und beginnt wieder bei 0. Dies kann korrigiert werden, indem man zum gemessenen Wert 360° addiert.

Nebenstehendes Programm adressiert diese beiden Punkte.

Winkelgeschwindigkeit3.py 1/34

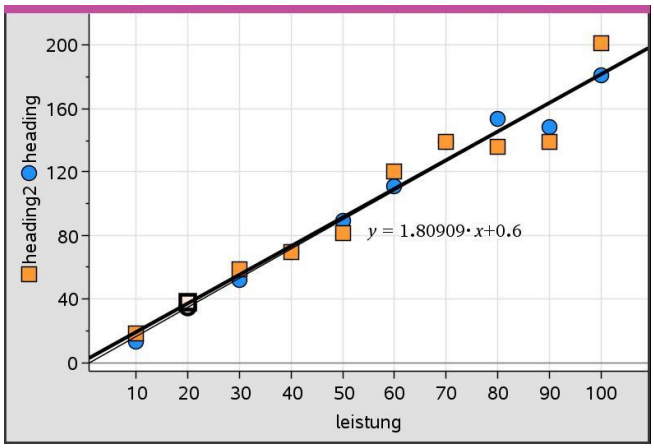
```
from codrone_edu import *

dauer = 1 # Ausführungsdauer in Sekunden
drone=Drone()
drone.pair()
drone.takeoff()
drone.hover(2)
headings = []
leistung = []
for i in range(10):  # Heading in Abhängigkeit von Leistung messen
    drone.set_yaw((i+1)*10) # Leistung in 10er-Schritten von 10% bis 100%
    drone.move(dauer)
    headings.append(drone.get_heading())
    leistung.append((i+1)*10)
drone.land()
# Umrechnung der Heading-Werte in absolute Winkel-Änderungen
headings_delta = []
headings_delta.append(headings[0])

for i in range(1, len(headings)):
        current = headings[i]
        previous = headings[i-1]
        # Absolute Änderung berechnen
        delta = current - previous

        # Korrektur des Überlaufs:
        if delta < 0:
            delta += 360
        headings_delta.append(delta)

store_list("heading2",headings_delta)
store_list("Leistung2",leistung)
drone.close()
```

Ergänzen Sie die Messwerte des Flugs ohne Zwischenlandung im Lists & Spreadsheet-Fenster und führen Sie eine zweite Y-Variable im Data & Statistics-Fenster ein.	

v. Reflektion, Gelerntes

Vergleichen Sie Ihre Messergebnisse und Erkenntnisse mit den anderen Teilnehmern in der Gruppe.

Die Auswertung der Winkelgeschwindigkeit in Abhängigkeit von der Leistung zeigt ein lineares Verhältnis von ungefähr 1,8° pro %-Leistung. Bei einer Leistung von 100 % dreht sich die CoDrone mit ungefähr 180°/s. Diesen Messwert brauchen wir später nochmal, im Modul 3.
Bei Leistungen unterhalb 50 % ist die Messwert-Streuung noch relativ gering, nimmt aber bei höherer Leistung deutlich zu. Dies gilt es bei der Programmierung von Bewegungsvektor-basierten Flügen zu berücksichtigen. Wenn es auf Präzision der Flugmission ankommt, sollte man unterhalb 50 % Leistung bleiben.
Speziell in der freiwilligen Zusatzübung (3) wurde gelernt, dass Roh-Messwerte manchmal nicht direkt ausgewertet werden können, sondern erst durch mathematische Umrechnung in ein sinnvolles Format gebracht werden müssen.

vi. Anregungen und Erweiterungen

Wiederholen Sie die Messreihe bei einer größeren Flughöhe und über einem anderen Untergrund.
Ändern und erweitern Sie ihren Python Code und nehmen Sie eine weitere Messreihe zur Erfassung der Geschwindigkeit der CoDrone im Vorwärtsflug auf.

© Texas Instruments EMEA Sales GmbH 2026

3. Modul 2 – Messwerterfassung mit der CoDrone

3.1 Höhenmessung oder Luftdruckmessung?

i. Hintergrund und Aufgabe

In der bemannten und unbemannten Luftfahrt unterscheidet man zwischen zwei verschiedenen Referenzpunkten bei Höhenangaben:
1. MSL (Mean Sea Level): Die nautische Höhe. Sie bezieht sich auf die Meereshöhe, auch N.N. genannt. Diese Höhe wird von bemannten und unbemannten Fluggeräten mit einem Barometer-Sensor gemessen und für Flugrouten und Korridore verwendet.
2. AGL (Above Ground Level): Dies ist die auf den Boden unter dem Fluggerät bezogene Höhe. Sie ändert sich mit dem Bodenprofil.
Ein Beispiel:
Eine Drohne startet und fliegt auf 40 m Höhe. Danach fliegt sie über ein 15 m hohes Objekt. Die Drohne bleibt dabei auf 40 m, aber der Abstand zum Objekt beträgt jetzt nur noch 25 m.
Ähnlich verhält es sich mit der CoDrone. Sie besitzt einen barometrischen Sensor zur Bestimmung ihrer Flughöhe und einen Abstandssensor zur Bestimmung der Entfernung zum Boden. Beim Schweben über dem Startplatz sind beide, Flughöhe und Abstand zum Boden gleich und der Luftdruckwert wird mit dem Abstandssensor zum Boden kalibriert.
Wir machen einen Messwerterfassungs-Flug in dieser Übung und schauen uns die mathematische Beziehung dieser beiden Größen näher an.

Schreiben Sie ein Python-Programm, mit dem die Messwerte des barometrischen und des Abstandssensor der CoDrone bei verschiedenen Flughöhen gemessen, gespeichert und ausgewertet werden.

ii. Lernziel

Erlernen eines computergestützten Messwerterfassungsprozesses:
1. Planen der Messreihe
2. Code-Erstellung
3. Durchführung der Messreihe und Speichern der Messwerte
4. Auswertung der Messreihe
5. Schlussfolgerungen, mathematische und physikalische Hintergründe

Anwendung von kommando-basierten Flugmanövern, Speichern der in Python gewonnenen Messdaten in TI-Nspire™ -Listen, Darstellung und Analyse der Messdaten in Lists & Spreadsheet- und Data & Statistics-Fenstern in der TI-Nspire™ Umgebung.

© Texas Instruments EMEA Sales GmbH 2026

iii. Zielgruppe

Schüler und Schülerinnen ab der 7. Jahrgangsstufe mit Vorkenntnissen in einer Programmiersprache, vorzugsweise Python.

iv. Anleitung und Code

Sichern Sie sich einen Flugraum von 100 x 100 x 200 cm, in dem gefahrlos und ohne Hindernisse beliebige Flugmanöver geflogen werden kann.

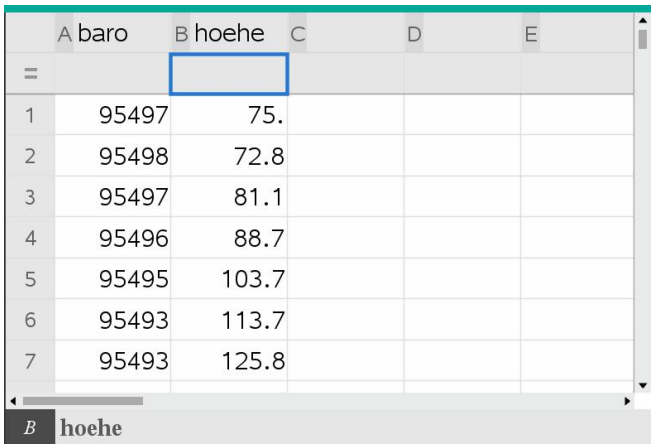
Schritte	Python Code-Fenster
1. Erzeugen Sie eine neue TI-Nspire™ -Datei und nennen Sie sie „Modul2“. Fügen Sie ein neues Python Fenster ein und vergeben Sie einen sinnvollen Namen, zum Beispiel „Hoehenmessung“. Speichern Sie Ihr Projekt regelmäßig, um Datenverlust zu vermeiden.	

2. Erstellen Sie Python Code zur Messwerterfassung von Barometer-Daten und Abstandssensor-Daten. Beginnen Sie mit der Flughöhe nach takeoff() und erhöhen Sie die Flughöhe nach jeder Messdatenerfassung um 10 cm. Stellen Sie sicher, dass Sie die im Flugraum maximal zulässige Flughöhe nicht überschreiten. Speichern Sie die Messwerte in Python Listen. Nach der Messwerterfassung landen Sie die Drohne und kopieren die Messwerte aus den Python Listen in Nspire Listen zur weiteren Messdatenverarbeitung.
 	
Über das Menu des TI-Nspire™ CX II wählen Sie alle nötigen Kommandozeilen aus.
Menu →
Weitere Module →
Robolink Codrone EDU →
from codrone_edu import *
Sensors →
	Hoehenmessung.py
	`from codrone_edu import *`
	`drone=Drone()`
	`drone.pair()`
	`baro = []`
	`hoehe = []`
	`drone.takeoff() # Drohne schwebt ca 80cm über Grund`
	`for i in range(12): # 12 10cm Schritte bis 200cm Höhe`
	`  baro.append(drone.get_pressure())`
	`  hoehe.append(drone.get_height())`
	`  drone.move_up(10)`
	`baro.append(drone.get_pressure()) # Letzter Messwert bei 200cm`
	`hoehe.append(drone.get_height())`
	`drone.land()`
	`store_list("Baro",baro)`
	`store_list("Hoehe",hoehe)`
	`drone.close()`

© Texas Instruments EMEA Sales GmbH 2026

3. Ergänzen Sie ein Lists & Spreadsheet-Fenster und verknüpfen Sie zwei Spalten mit den zuvor im Python Code definierten Variablen.

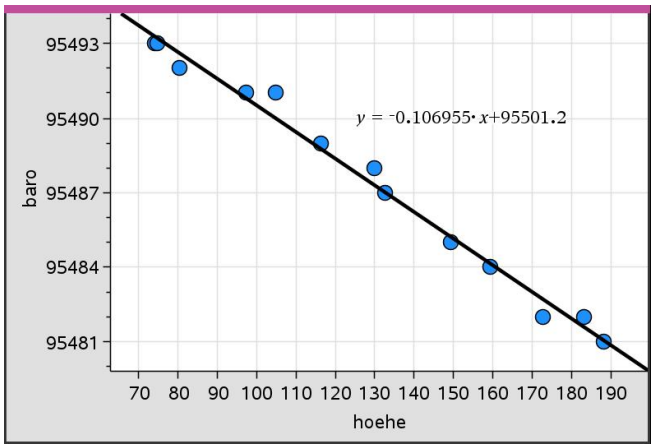
Var (Taste drücken) →
Verknüpfen mit: →
Baro
Var (Taste drücken) →
Verknüpfen mit: →
Hoehe



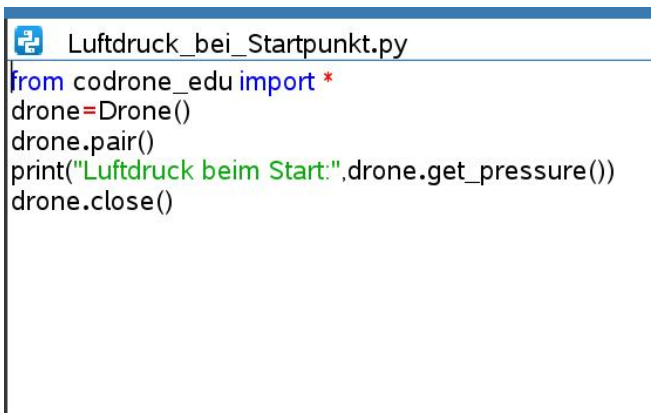
4. Fügen Sie ein Data & Statistics-Fenster ein, um die Messreihe graphisch anzuzeigen. Verknüpfen Sie die x- und y-Achsen mit den entsprechenden Nspire Variablen. Justieren Sie den Zoom auf die vorhandenen Daten. Ergänzen Sie eine lineare Regression über die Messwerte.

Rechter Mausklick im Bereich links neben der y-Achse: Zoom → Zoom – Daten

Menu → Analysieren → Regression →
Lineare Regression (y = mx + b)



5. Diskutieren Sie die Ergebnisse:
Aus der Regressionsgleichung lässt sich der Luftdruck am Boden zum Zeitpunkt des Experiments theoretisch bestimmen. Überprüfen Sie dies praktisch, indem Sie beiliegendes Programm verwenden.
Aus der Regressionsgleichung lässt sich die Beziehung zwischen Luftdruck und Flughöhe bestimmen.



v. Reflektion, Gelerntes

Mit dieser Übung haben Sie den kompletten Messwerterfassungs-Prozess durchgeführt. In der Analyse haben Sie mathematische Hilfsmittel (lineare Regression, Ermitteln der Regressionsgleichung) verwendet. Durch Extrapolation haben Sie den Luftdruck bei Start des Experiments am Boden ermittelt und durch

© Texas Instruments EMEA Sales GmbH 2026

eine praktische Messung bestätigt.

vi. Anregungen und Erweiterungen

Wiederholen Sie die Messreihe, um ein Gefühl über die Reproduzierbarkeit der Messreihe zu gewinnen. Ergänzen sie in der Mess-Schleife nach jeder Höhenänderung ein drone.hover(1) Statement, um die Drohne 1 Sekunde lang auf der neuen Höhe schweben zu lassen, bevor die Druck- und Höhenmessung stattfindet. Wird die Messung dadurch genauer? Welche Rückschlüsse können Sie aufgrund des hier im Beispiel gemessenen Luftdrucks auf die Höhe des Wohnorts (über N.N.) des Autors dieses Workshops ziehen?
(Hinweis: Standarddruck auf Meereshöhe: 1013,25 hPa)

© Texas Instruments EMEA Sales GmbH 2026

3.2 Untersuchung der Restflugdauer der CoDrone in Abhängigkeit des Akkuladezustands

i. Hintergrund und Aufgabe

Man kennt es vom e-Bike, Elektro-Auto oder Verbrenner-Auto: Der Bordcomputer gibt Auskunft über die mögliche Reichweite des Fahrzeugs auf Basis des aktuellen Akkuladezustands oder des Füllstands im Kraftstofftank. So etwas wäre auch für die CoDrone hilfreich. Bitte beachten Sie, dass die TI-Nspire™ CoDrone Firmware bereits bei 50 % Akkuladung den Flug der Drohne aus Sicherheitsgründen abbricht. Manuell könnte man bei Gebrauch der CoDrone Steuerung noch deutlich länger fliegen.

In dieser Aufgabe soll die Abhängigkeit der Flugdauer vom Akkuladezustand näher untersucht werden. Schreiben Sie ein Python-Programm, welches in festen Zeitabständen die Akkuladung der Drohne im Flugbetrieb misst und für die spätere Auswertung speichert. Analysieren Sie die Daten und ermitteln Sie eine Regressionsfunktion der Entladekurve. Erstellen Sie eine mathematische Formel zur Ermittlung der Restflugdauer bis zum 50 % Akkuladungspunkt.

ii. Lernziel

Vertiefen des computergestützten Messwerterfassungsprozesses:

1. Planen der Messreihe
2. Code-Erstellung
3. Durchführung der Messreihe und Speichern der Messwerte
4. Auswertung der Messreihe
5. Erstellung der mathematischen Beziehung der Messgrößen
6. Überprüfung der Hypothese im Praxis-Experiment

Anwendung von kommando-basierten Flugmanövern, Speichern der in Python gewonnenen Messdaten in TI-Nspire™ Listen, Darstellung und Analyse der Messdaten in Lists & Spreadsheet- und Data & Statistics-Fenstern in der TI-Nspire™ Umgebung.

iii. Zielgruppe

Schüler und Schülerinnen ab der 7. Jahrgangsstufe mit Vorkenntnissen in einer Programmiersprache, vorzugsweise Python.

© Texas Instruments EMEA Sales GmbH 2026

iv. Anleitung und Code

Sichern Sie sich einen Flugraum von 100 x 100 x 200 cm, in dem gefahrlos und ohne Hindernisse beliebige Flugmanöver geflogen werden kann. Stellen Sie sicher, dass der Akku der CoDrone vor dem Experiment zu 100 % vollgeladen ist.

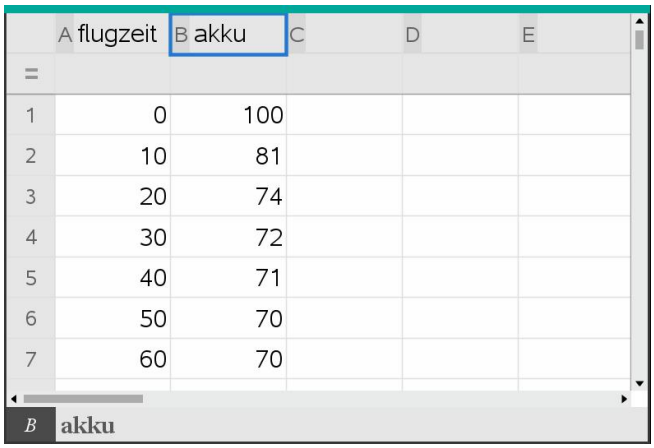
Schritte	Python Code-Fenster
1. Benutzen Sie Ihre bestehende TI-Nspire™ - Datei „Modul2“ und fügen Sie ein neues Python Fenster ein, mit dem Namen „Restflugzeit_Erfassung“. Speichern Sie Ihr Projekt regelmäßig, um Datenverlust zu vermeiden.	

2. Erstellen Sie Python Code zur Messwerterfassung von Akkuladungs-Daten und Flugzeit-Daten. **Achtung:** Die TI-Nspire™ Firmware landet die Drohne automatisch bei einem Akkustand niedriger als 50 % und bricht das Programm ab. Um diesen Effekt zu vermeiden und das Programm ordnungsgemäß zu beenden lassen Sie die Drohne nur bis zu einer Akkuladung von 55 % schweben und danach sofort landen. Messen Sie alle 10 Sekunden die Akkuladung und speichern den Wert zusammen mit der Flugzeit in Python Listen. Ist die Akkuladung unter 55 % gesunken, landen Sie die Drohne und kopieren die Messwerte aus den Python Listen in Nspire Listen zur weiteren Messdatenverarbeitung.
 	
Über das Menu des TI-Nspire™ CX II wählen Sie alle nötigen Kommandozeilen aus.
Menu →
Weitere Module →
Robolink Codrone EDU →
from codrone_edu import *
Sensors →
	Restflugzeit_Erfassung.py 19/25
	`from codrone_edu import *`
	`from time import time`
	`drone=Drone()`
	`drone.pair()`
	`flugdauer = []`
	`akkuladung = []`
	`restladung = 55 # Programmabbruch vermeiden!!`
	`messrate = 10 # Messwerterfassung alle 10 Sekunden`
	`elapsed = 0`
	`tick = time()`
	`bat = drone.get_battery()`
	`flugdauer.append(elapsed)`
	`akkuladung.append(bat)`
	`drone.takeoff()`
	`while bat > restladung:`
	`  if time() > tick + messrate:`
	`  elapsed += messrate`
	`  bat = drone.get_battery()`
	`  akkuladung.append(bat)`
	`  flugdauer.append(elapsed)`
	`  tick = time()`
	`drone.land()`
	`store_list("Flugzeit",flugdauer)`
	`store_list("Akku",akkuladung)`
	`drone.close()`

3. Fügen Sie ein Lists & Spreadsheet-Fenster hinzu und verknüpfen Sie zwei Spalten mit den zuvor im Python Code definierten Variablen.

Spalte A selektieren
Var (Taste drücken) →
Verknüpfen mit: →
flugzeit

Spalte B selektieren
Var (Taste drücken) →
Verknüpfen mit: →
akku

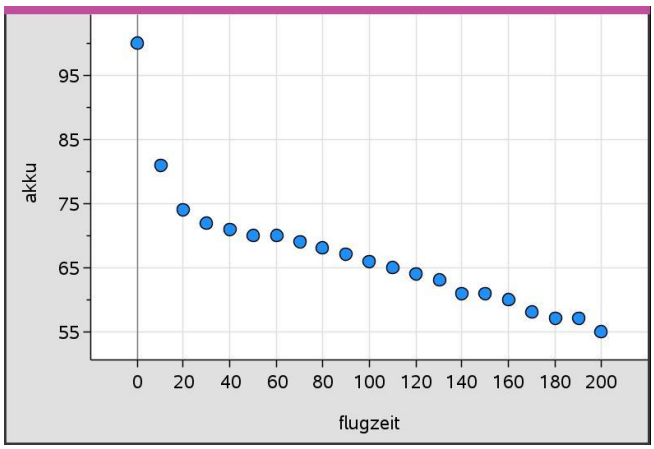


	A flugzeit	B akku	C	D	E
1	0	100			
2	10	81			
3	20	74			
4	30	72			
5	40	71			
6	50	70			
7	60	70			

B akku

4. Fügen Sie ein Data & Statistics-Fenster ein, um die Messreihe graphisch anzuzeigen. Verknüpfen Sie die x- und y-Achsen mit den entsprechenden Nspire Variablen. Justieren Sie den Zoom auf die vorhandenen Daten.

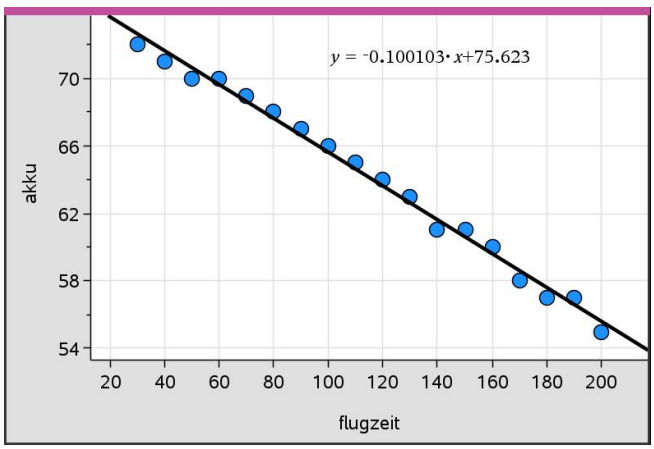
Rechter Mausklick im Bereich links neben der
y-Achse: Zoom → Zoom – Daten



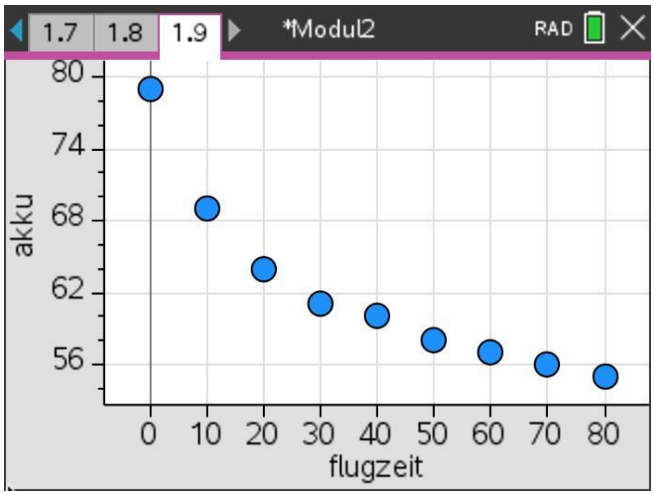
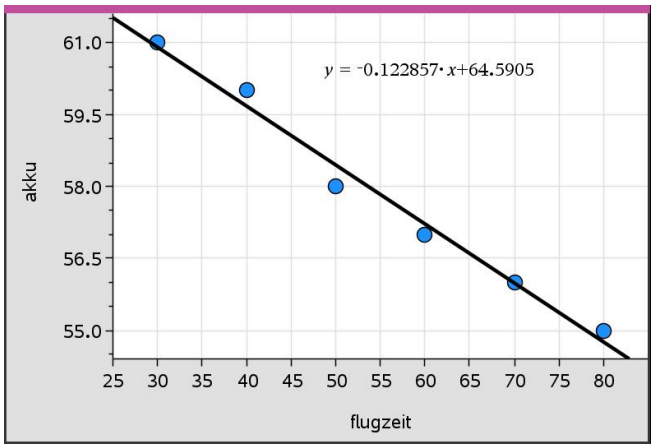
5. Diskutieren Sie die Ergebnisse:
Für die ersten 20 Sekunden nach dem Start verhält sich die Akku-Ladungsanzeige stark nichtlinear. Erst nach 20-30 Sekunden Flugzeit lässt sich eine hinreichend lineare Beziehung zwischen Akkuladung und Flugzeit herleiten. Entfernen Sie die Messwerte der ersten 20 Sekunden Flugzeit aus dem Spreadsheet. Fügen Sie eine lineare Regression ein.

Menu → Analysieren → Regression →
Lineare Regression (y = mx + b)

Aus der Regressionsgleichung lässt sich ablesen, dass im linearen Bereich der Akkuladungsanzeige 1 % Akkuladung in etwa 10 Sekunden Flugdauer bereitstellen.



© Texas Instruments EMEA Sales GmbH 2026

6. Hypothese: Die Akkuladungs-Messung zeigt in den ersten 20 Sekunden nach Start der Drohne stark nichtlineares Verhalten. Zum Beweis nehmen Sie einen teilentladenen Akku mit etwa 75 % Ladung bei stillstehender Drohne und wiederholen das Experiment. Sollte die Hypothese richtig sein, müsste die Akkuladung in den ersten 20 Sekunden nach Start der Drohne wieder stark absinken, um sich danach wieder um den Wert von 1 % pro 10 Sekunden Flugzeit einzupendeln. Analysieren Sie die neu aufgenommenen Messwerte. Man erkennt wieder das stark nichtlineare Verhalten der Akkuladungsanzeige während der ersten 20 Sekunden des Fluges.	
7. Entfernen Sie die Messwerte der ersten 20 Sekunden und fügen eine Regressionsgerade ein. Ähnliches Ergebnis wie beim vollgeladenen Akku: Im linearen Bereich der Ladungsanzeige beträgt die Flugzeit bei teilentladenem Akku etwa 8 Sekunden pro % Akkuladung.	

v. Reflektion, Gelerntes

Mit dieser Übung haben Sie erkannt, dass die Akkuladungsanzeige der CoDrone keine echte Coulomb-Anzeige ist, sondern auf einer Näherungsformel beruht, wahrscheinlich basierend auf Messung der Akkuspannung. Diese scheint im Leerlauf und in den ersten 20 Sekunden nach Beginn des Drohnenflugs nichtlinear abzusinken, bevor sie ein lineares Verhalten zwischen Flugdauer und Akkuladung zeigt. Die Realisierung einer Restflugdauer-Anzeige erscheint daher nicht sinnvoll, zumal in vielen der in diesem Workshop gezeigten Experimente die Flugdauer unter 30 Sekunden beträgt.

Der Akkustand wird am Anfang eines Flugs oft zu hoch angezeigt.

Das heißt: Wenn die Akkuladung zu Beginn weniger als 65 % beträgt, kann die Mission abgebrochen werden, weil die Grenze von 50 % schon vorher erreicht wird. Deshalb sollte man immer mit einem vollgeladenen Akku starten – besonders bei Flügen, die länger als 30 Sekunden dauern.

Mit einem vollen Akku sind etwa 5 Minuten Flugzeit möglich, da die Firmware eine Mindestgrenze von über 50 % vorgibt.

Die durch die TI-Nspire™ Firmware festgelegte Mindestgrenze von 50 % Ladezustand ermöglicht bei vollgeladenem Akku eine Flugzeit von knapp 5 Minuten.

© Texas Instruments EMEA Sales GmbH 2026

vi. Anregungen und Erweiterungen

Ermitteln Sie experimentell die Restflugdauer der CoDrone, nachdem die Akkuladung von 50 % im Flugbetrieb unterschritten wurde. Legen Sie eine Stoppuhr bereit und trennen Sie die Fernsteuerung vom PC. Starten Sie die Drohne manuell mit der Fernsteuerung, lassen Sie sie auf etwa 80 cm schweben und stoppen Sie die Zwischen-Zeit (Lap), bis die Meldung „Batterie niedrig“ kommt. Lassen Sie die Drohne weiter schweben, bis sie automatisch landet. Stoppen Sie die Gesamtzeit. Hier meine gemessenen Werte als Referenz: 5:00 Minuten bis zur „Batterie niedrig“ Warnung, weitere 30 Sekunden bis zur automatischen Landung.

© Texas Instruments EMEA Sales GmbH 2026

3.3 Interaktive Objekterkennung mit der Drohne

i. Hintergrund und Aufgabe

Wir haben in Übung (2.1) gelernt, dass die CoDrone einen Sensor zum Messen des Abstands zum Boden besitzt, ihre Flughöhe jedoch barometrisch misst. Wenn man die zu untersuchende Fläche in gleicher (barometrischer) Höhe abfliegt und den Abstand zum Boden mit dem Entfernungssensor misst, kann auf diese Weise ein Objekt in der Fläche gefunden werden. Die Steuerung der Flugbahn der Drohne soll interaktiv erfolgen, per Tastendruck auf dem TI-Nspire™ CX II Handheld.
Schreiben Sie ein Python-Programm, welches die Drohne über die Cursor-Tasten des TI-Nspire™ CX II Handhelds steuert. Messen Sie nach jeder Flugbewegung den Abstand zum Boden und entscheiden Sie auf Basis des Messergebnisses, ob ein auf dem Boden liegendes Objekt entdeckt wurde. Zeigen Sie auf dem Handheld die Koordinaten vom Fundort an.

ii. Lernziel

Interaktive Steuerung der Drohne über die Tastatur des TI-Nspire™ CX II Handhelds. Objekterkennung auf dem Boden über den Abstandssensor der Drohne.

iii. Zielgruppe

Schüler und Schülerinnen ab der 7. Jahrgangsstufe mit Vorkenntnissen in einer Programmiersprache, vorzugsweise Python.

© Texas Instruments EMEA Sales GmbH 2026

iv. Anleitung und Code

Sichern Sie sich einen Flugraum von 200 x 200 x 200 cm, in dem gefahrlos und ohne Hindernisse beliebige Flugmanöver geflogen werden kann. Stellen Sie sicher, dass der Akku der CoDrone vor dem Experiment zu 100 % vollgeladen ist. Markieren Sie den Startpunkt der Drohne, um das Experiment eventuell zu wiederholen. Platzieren Sie ein ausreichend (> 8 cm) hohes, aber dennoch unterhalb der Flughöhe (ca. 70 cm) liegendes Objekt im Bereich des Flugraums.

Schritte	Python Code-Fenster

1. Untersuchen Sie die Tastaturabfrage-Funktion get_key() vom TI-Nspire™ CX II. Sie finden die Funktion in der ti_system Blibliothek.
Menu →
Weitere Module →
TI System →
from ti_system import get_key
 	
Die get_key() – Funktion gibt einen String der gedrückten Taste zurück, z. B. Drücken der <esc> Taste des Handhelds wird als String „esc“ zurückgegeben.
get_key(0) wartet nicht auf das Drücken einer Taste, sondern gibt entweder einen leeren String „“ oder den im Tastaturpuffer enthaltenen String zurück. get_key(1) hingegen hält das Programm an und wartet, bis eine Taste gedrückt wurde. Drücken Sie nacheinander die Cursor-Tasten des Handhelds und notieren Sie sich, welche Strings von den jeweiligen Tasten zurückgegeben wurden. Drücken Sie die <esc>-Taste des Handhelds, um das Programm zu beenden.
	Tastenerkennung.py 2/9
	`from ti_system import get_key`
	
	`a = ""`
	`while a != "esc":`
	`  print("Drücke eine Taste:")`
	`  a = get_key(1)`
	`  print(a," erkannt.")`

2. Erstellen Sie einen Entscheidungsbaum für die Steuerung der CoDrone. Verwenden Sie hierfür den Python elif – Befehl und platzieren Sie print-Befehle zum Ausprobieren Ihres Entscheidungsbaums. Auch hier beenden Sie ihr Programm mit <esc>.
 	
Menu →
Eingebaute →
Steuerung →
elif:
	Steuerung.py 1/18
	`from ti_system import get_key`
	
	`a = ""`
	`print("Drohne ist gestartet.")`
	`print("Steuerung starten!")`
	`while a != "esc":`
	`  a = get_key(1)`
	`  if a == "up":`
	`  print("Drohne fliegt 5cm vorwärts.")`
	`  elif a== "down":`
	`  print("Drohne fliegt 5cm rückwärts.")`
	`  elif a== "left":`
	`  print("Drohne fliegt 5cm nach links.")`
	`  elif a== "right":`
	`  print("Drohne fliegt 5cm nach rechts.")`
	`print("Drohne landet.")`

© Texas Instruments EMEA Sales GmbH 2026

3. Speichern Sie sich eine Kopie Ihres Steuerungscodes in einem neuen Python-Fenster ab, ergänzen ihn durch den üblichen Python-Code zum Initialisieren und Starten der CoDrone. Ergänzen Sie weitere Variablen zum Speichern der Positionsänderungen der CoDrone und ersetzen Sie die print()-Befehle durch die entsprechenden CoDrone-Bewegungs- und Sensor-Funktionen wie nebenstehend gezeigt.
Die <from ti_system import get_key> Zeile können Sie löschen, die get_key()-Funktion ist auch Teil des codrone_edu Moduls.
Die Variable <schritt> legt die verwendete „Schrittweite“ der Drohnen-Flugbefehle fest. Hier im Beispiel wurde 5 cm gewählt. Sie können die Schrittweite auch auf z. B. 10 cm vergrößern. Allerdings sollte die Schrittweite klein genug gewählt werden, um das Objekt auch überfliegen zu können (Faustregel: schritt ≤ kleinstes Maß(x,y) des Objekts / 2).

Objekterkennung.py 19/35

```
from codrone_edu import *
drone=Drone()
drone.pair()
drone.set_feedback("hide")
drone.takeoff()
a = ""
schritt = 5
print("Steuerung starten!")
x = 0
y = 0
while a != "esc":
    a = get_key(1)
    h0 = int(drone.get_height())
    if a == "up":
        drone.move_forward(schritt)
        x += schritt
    elif a== "down":
        drone.move_backward(schritt)
        x -= schritt
    elif a== "left":
        drone.move_left(schritt)
        y += schritt
    elif a== "right":
        drone.move_right(schritt)
        y -= schritt
    h1 = int(drone.get_height())
    print("h0:",h0,"h1:",h1,"x:",x,"y:",y)
print("Drohne landet.")
drone.land()
t,x1,y1,z1=drone.get_position_data()
print("Flugdauer:",int(t),"Sekunden")
print("Landeposition x1:",x1,"y1:",y1)
drone.close()
```

4. Machen Sie einen Probeflug und fliegen nach dem Start der CoDrone über das Objekt. Vermeiden Sie langen Tastendruck, da die repeat-Funktion des Handhelds oder des PCs den Zeichenpuffer der get_key()-Funktion füllt und das Verhalten der Drohne dann träge wird bzw. zu viele move()-Befehle in eine Richtung gesendet werden.
Beobachten Sie das Verhalten der CoDrone, insbesondere zu Beginn des Überflugs des Objekts und danach wieder nach Verlassen des Objektbereichs.
Erkennen Sie anhand des Messprotokolls in der Shell, wann die Drohne über dem Objekt geschwebt hat und wann sie den Objektbereich wieder verlassen hat?
Vergleichen Sie die gespeicherten Schritt-Koordinaten-Werte (x, y) mit den Koordinaten-Werten (x1, y1) des CoDrone-Sensors nach der Landung.

Python-Shell 20/39

```
Steuerung starten!
h0: 70 h1: 71 x: 0 y: 5
h0: 71 h1: 71 x: 0 y: 10
h0: 68 h1: 73 x: 0 y: 15
h0: 72 h1: 73 x: 0 y: 20
h0: 74 h1: 73 x: 0 y: 25
h0: 74 h1: 74 x: 0 y: 30
h0: 74 h1: 71 x: 0 y: 35
h0: 70 h1: 81 x: 0 y: 40
h0: 76 h1: 75 x: 0 y: 45
h0: 80 h1: 77 x: -5 y: 45
h0: 77 h1: 72 x: -10 y: 45
h0: 71 h1: 69 x: -15 y: 45
h0: 69 h1: 71 x: -20 y: 45
h0: 75 h1: 79 x: -25 y: 45
h0: 79 h1: 81 x: -30 y: 45
h0: 83 h1: 80 x: -35 y: 45
h0: 77 h1: 73 x: -40 y: 45
h0: 72 h1: 69 x: -45 y: 45
h0: 67 h1: 69 x: -45 y: 40
h0: 69 h1: 72 x: -45 y: 35
h0: 74 h1: 73 x: -45 y: 30
h0: 74 h1: 75 x: -45 y: 25
h0: 73 h1: 75 x: -45 y: 20
h0: 76 h1: 75 x: -40 y: 20
h0: 73 h1: 75 x: -35 y: 20
h0: 74 h1: 75 x: -30 y: 20
h0: 74 h1: 73 x: -25 y: 20
h0: 73 h1: 72 x: -20 y: 20
h0: 74 h1: 72 x: -25 y: 20
h0: 74 h1: 73 x: -30 y: 20
h0: 71 h1: 72 x: -25 y: 20
h0: 71 h1: 72 x: -25 y: 20
Drohne landet.
Flugdauer: 47 Sekunden
Landeposition x1: -25 y1: 22
>>>
```

© Texas Instruments EMEA Sales GmbH 2026

v. Reflektion, Gelerntes

Sie haben die get_key()-Funktion des TI-Nspire™ -Systems kennengelernt, eine interaktive Programmsteuerung aufgebaut und damit die Flugbewegungen und die Landung der CoDrone gesteuert. Sie haben gesehen, dass die CoDrone beim Überfliegen des Objekts die Flughöhe anpasst, d. h. höher fliegt. Dies deutet darauf hin, dass die CoDrone ihre Flughöhe durch beide Sensoren, Barometer und Höhensensor, kontrolliert und anpasst. Sie haben gelernt, dass die x1- und y1-Koordinaten des Bewegungs-Sensors erstaunlich gut mit dem kumulativen Ergebnis aus den Steuerungsbefehlen übereinstimmen.

vi. Anregungen und Erweiterungen

Versuchen Sie eine mathematische Beziehung zwischen den Höhenmessungen in der Messreihe zu finden, um eine Objekterkennung im Flug aufgrund der Höhenmessungen der CoDrone durchzuführen. Wiederholen Sie ein paar Flugmissionen und überprüfen Sie die Robustheit Ihrer Objekterkennungs-Formel. Gegebenenfalls ziehen Sie die Messdaten des Barometer-Sensors hinzu, indem Sie einen Messwert nach dem takeoff()-Kommando als Referenz abspeichern und für Ihre Berechnungen zur Objekterkennung als Vergleich hinzuziehen.

© Texas Instruments EMEA Sales GmbH 2026

4. Modul 3 – komplexe Flugmanöver

4.1 Physikalische und mathematische Grundlagen

Die CoDrone gehört zum Typ der Quadcopter. Alle Quadcopter besitzen vier Rotoren, im Gegensatz zu Tricoptern (3 Rotoren), Hexacoptern (6 Rotoren) oder Octocoptern (8 Rotoren).

Die Physik des Quadcopter-Fliegens
Um zu verstehen, wie eine Drohne fliegt, nutzen wir das **3. Newton'sche Gesetz** (Axiom der Wechselwirkung): *„Kräfte treten immer paarweise auf. Übt Körper A eine Kraft auf Körper B aus, wirkt eine gleich große, aber entgegengesetzte Kraft von Körper B auf Körper A."*[4]

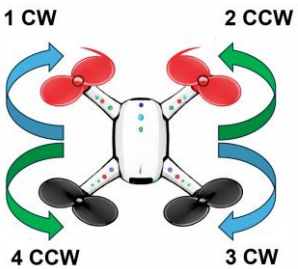
1. Das Schweben (Kräftegleichgewicht)
Damit die CoDrone abhebt und schwebt, müssen zwei Kräfte im Gleichgewicht sein:

- **Gewichtskraft ($\overrightarrow{F_g}$):** Die Erdanziehungskraft zieht die Drohne nach unten ($\overrightarrow{F_g} = m_{CoDrone} * \vec{g}$).
- **Schubkraft ($\overrightarrow{F_s}$):** Die Rotoren drücken die Luft nach unten (**Actio**). Als Reaktion drückt die Luft die Drohne mit der gleichen Kraft nach oben (**Reactio**).

Merksatz: Wenn die Summe der Schubkräfte aller vier Rotoren ($\sum \overrightarrow{Fs}$) genau so groß ist wie die Gewichtskraft ($\overrightarrow{Fg}$), schwebt die Drohne in der Luft.

2. Die Drehung (Das Drehmoment)
Ein Motor erzeugt beim Drehen des Rotors ein **Drehmoment**. Würden sich alle vier Rotoren in dieselbe Richtung drehen, würde sich die Drohne als Gegenreaktion (3. Newton'sche Gesetz) unkontrolliert in die andere Richtung um die Hochachse drehen.

Die Lösung:



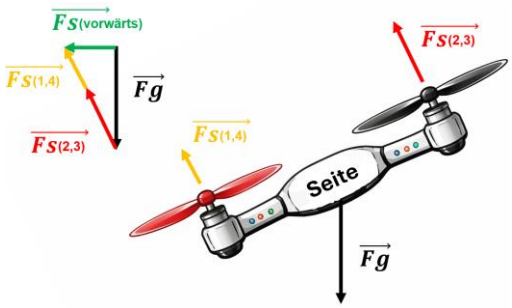
- Zwei Rotoren drehen sich **im Uhrzeigersinn** (CW).
- Zwei Rotoren drehen sich **gegen den Uhrzeigersinn** (CCW).
- Die entstehenden Drehmomente sind entgegengesetzt gerichtet und heben sich in ihrer Wirkung auf die Drohne auf. Die Drohne bleibt stabil.

Eine kontrollierte (gewollte) Drehung der Drohne um die Hochachse wird durch unterschiedliche Drehzahlen der Rotorpaare erzeugt. Die

[4] van Hees, H. (2025). Newton'sche Mechanik. In: Mechanik für das gymnasiale Lehramt. Springer Spektrum, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-69047-5_2

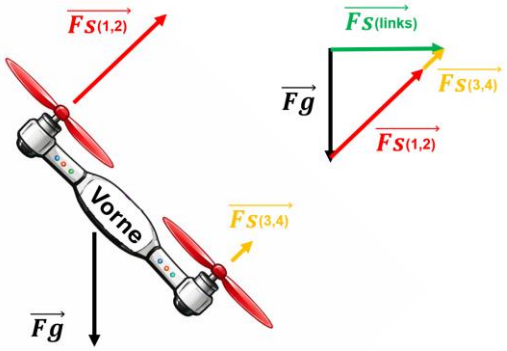
Drehmomente der Rotoren heben sich nur teilweise auf und die Drohne dreht sich im oder gegen den Uhrzeigersinn um die Hochachse.

3. Das Fliegen nach vorne



Drehen sich die vorderen Rotoren der Drohne langsamer als die hinteren, ist die Schubkraft der hinteren Rotoren größer und die Drohne macht eine Nick-Bewegung nach vorne. Die Kraftrichtung der Schubkraft ist also nicht länger senkrecht zur Erdanziehungskraft, sondern teilt sich gemäß Kräfteparallelogramm auf in eine Hub-Komponente (Ausgleich der Schwerkraft) und eine Vortrieb-Komponente (Kraft nach vorne). Natürlich muss die gesamte Schubkraft $\sum \overrightarrow{F_S}$ in diesem Fall größer sein als beim Schweben, sonst würde die Drohne sinken.

4. Das Fliegen zur Seite



Drehen sich die linken Rotoren der Drohne langsamer als die rechten, macht die Drohne eine Roll-Bewegung nach links. Wieder wirken Kräfte als Kräfteparallelogramm, nur dieses Mal wirkt dieser Kraft-Vektor senkrecht zur Längsachse und erzeugt eine Seitwärtsbewegung.

5. Vektorbasierte Flugbewegung

In der CoDrone Firmware wurden Funktionen zur vektorbasierten Steuerung implementiert. Diese Funktionen können mit ihren Parametern vor-eingestellt werden und wirken wie Kraftvektoren in die Höhen-, Seiten-, Vorwärts/Rückwärts- und Drehbewegung der Drohne. Durch ein entsprechendes Kommando werden dann alle Funktionen gleichzeitig für eine definierte Zeit ausgeführt. Dies ermöglicht komplexe Flugmanöver, wie in der nachfolgenden Aufgabe illustriert.

© Texas Instruments EMEA Sales GmbH 2026

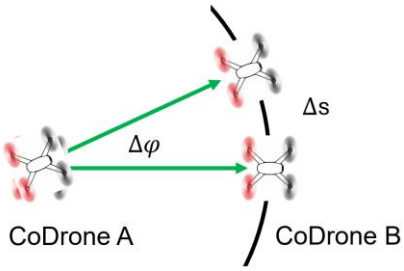
4.2 360° Selfie

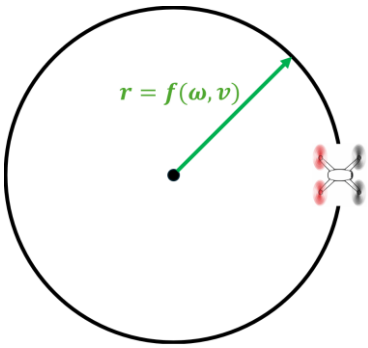
i. Hintergrund und Aufgabe

Moderne Kamera-Drohnen besitzen neben der klassischen Flugsteuerung über die Steuerknüppel einer Funk-Fernsteuerung auch voreingestellte Manöver, die von der Drohne automatisch ausgeführt werden. Ein vielfach verwendetes Manöver ist das 360 Grad Selfie. Dabei umfliegt die Drohne ein zuvor anvisiertes Objekt im Kreis. Dabei läuft die Kamera mit und erstellt während des Fluges ein Video vom Objekt von allen Seiten. Die CoDrone besitzt statt einer Front-Kamera einen Front-Abstandssensor.
Die Aufgabe besteht darin, mit Hilfe der vektorbasierten Flugbewegung ein imaginäres oder echtes Objekt in einer kreisförmigen Flugbahn zu umrunden und dabei die Drohne stets mit der Front zum Objekt zeigt.

Vorüberlegungen

Für die kreisförmige Umrundung werden drei Bewegungsvektoren benötigt, die sich gegenseitig überlagern:
a) Drehbewegung (CoDrone A): In Übung (1.3) haben wir gelernt, dass die CoDrone sich mit einer Winkelgeschwindigkeit von 1,8°/s pro % Leistung um die Hochachse dreht. Bei einer Leistung von 50 % braucht die CoDrone demnach 4 Sekunden für eine 360° Drehung. Für die 360° Drehung auf einer Kreisbahn brauchen wir noch Überlagerungen mit weiteren Bewegungsvektoren.





b) Seitwärtsbewegung: Die Geschwindigkeit in Seitwärtsrichtung ist ebenfalls proportional zur eingestellten Leistung. Wenn sich die Drohne gegen den Uhrzeigersinn um die Hochachse dreht, dann muss die Seitwärtsbewegung nach rechts erfolgen, um in der Überlagerung einen Flug auf einer Kreisbahn gegen den Uhrzeigersinn zu erzeugen. Je schneller die Drohne seitwärts fliegt, umso größer der Radius der Kreisbahn. Der Radius r ist eine Funktion der Winkelgeschwindigkeit ω und der Seitwärtsgeschwindigkeit v der Drohne.
c) Vorwärtsbewegung: Da die Drohne eine Masse m besitzt, wirkt eine Zentrifugalkraft auf die Drohne beim Flug auf der Kreisbahn. Es gilt: $F_{zf} = m * v^2/r$. Also brauchen wir noch einen dritten Vektor in Vorwärtsrichtung der Drohne, um der Zentrifugalkraft mit einer Zentripetalkraft F_{zp} der Drohne entgegenzuwirken und ein Kräftegleichgewicht in Radialrichtung herzustellen.

© Texas Instruments EMEA Sales GmbH 2026

ii. Lernziel

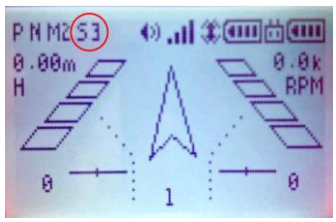
Kräfteparallelogramm und Anwendung des 3. Newton'schen Gesetzes. Tieferes physikalisches Verständnis von Flugbewegungen einer Quadcopter-Drohne. Experimentelle Untersuchung von Überlagerungen von Bewegungsvektoren. Räumliches Vorstellungsvermögen trainieren.

iii. Zielgruppe

Schüler und Schülerinnen ab der 11. Jahrgangsstufe mit Vorkenntnissen Physik (Mechanik) und in einer Programmiersprache, vorzugsweise Python.

iv. Anleitung und Code

Wichtig: Bevor Sie den CoDrone Flugcontroller mit dem TI-Nspire™ System verbinden, vergewissern Sie sich, dass der Controller auf Geschwindigkeitsstufe S3 steht. Gegebenenfalls drücken Sie wiederholt die L1-Taste des Controllers, bis „S3" im Display erscheint.



Schritte	Python Code-Fenster

1. Nebenstehend sehen Sie ein einfaches Beispiel-Programm für das 360° Selfie. Sollten Ihnen die Begriffe „roll, pitch und yaw" nicht geläufig sein, lesen Sie bitte im Abschnitt „CoDrone EDU Python Funktionsreferenz" des Einführungshandbuchs[5] die verwendeten Begriffe nach.
 	
Tasten Sie sich an die richtige Auswahl der drei Bewegungsparameter heran:
- Lassen Sie die Dreh-Leistung zunächst konstant bei 50 %.
- Lassen Sie die Drohne ruhig deutlich länger fliegen, als für eine 360° Runde nötig wäre. Dann sehen Sie leichter, ob die Drohne wirklich einen Kreis fliegt. Entfernt sie sich spiralförmig vom virtuellen Mittelpunkt, dann vergrößern Sie in kleinen Schritten die Zentrifugalkraft-Kompensation, bis die Drohne einen Kreis fliegt. Verkleinert die Drohne jedoch ihren Kreis-Radius während des Flugs, dann deutet dies auf eine zu starke Zentrifugal-Kompensation hin und Sie
	selfie_360.py 1/15
	`from codrone_edu import *`
	`drone=Drone()`
	`drone.pair()`
	`v_omega = 50# Drehung Hochachse, Leistung in %`
	`v_seite = 15 # Seiten-Geschwindigkeit, Leistung in %`
	`komp_zentrifugal = 15 # Kompensation, Leistung in %`
	`dauer = 16# Ausführungsdauer in Sekunden`
	`drone.takeoff()`
	`drone.set_yaw(v_omega)`
	`drone.set_roll(v_seite)`
	`drone.set_pitch(komp_zentrifugal)`
	`drone.move(dauer)`
	`drone.land()`
	`drone.close()`

[5] TI-Nspire™ CX II – CoDrone EDU Einführungshandbuch
https://education.ti.com/de/product-resources/codrone-edu

sollten komp_zentrifugal schrittweise verkleinern, bis die Drohne einen Kreis fliegt.

2. Eventuell wäre es hilfreich, wenn man bei einem Reihen-Experiment die eingestellten und gemessenen Werte für spätere Analysen speichern könnte. Dies wird mit nebenstehendem Code realisiert, indem alle relevanten Parameter nach der Flugmission in Nspire-Variablen gespeichert werden. Vergessen Sie nicht, auch Ihr Modul3.tns Projekt regelmäßig zu speichern, um auch in weiteren Workshop-Sitzungen wieder auf Ihre alten Experiment-Daten zugreifen zu können.

Die Daten werden mit der Funktion add_experiment_data() in Nspire-Variablen optional gespeichert. Diese Funktion ist der Übersichtlichkeit halber in einem eigenen Modul „selfie_archive_funcs“ gespeichert.

selfie_360_experimente.py 8/30

```
from codrone_edu import *
from time import sleep
from selfie_archive_funcs import *
drone=Drone()
drone.pair()
v_omega = 50# Drehung Hochachse, Leistung in %
v_seite = 25 # Seiten-Geschwindikeit, Leistung in %
komp_zentrifugal = 25 # Kompensation, Leistung in %
dauer = 12# Ausführungsdauer in Sekunden
r1 = drone.get_front_range()
drone.takeoff()
drone.set_yaw(v_omega)
drone.set_roll(v_seite)
drone.set_pitch(komp_zentrifugal)
drone.move(dauer)
delta = drone.get_heading()
if delta < 250: delta += 360
drone.land()
print("Drohne evt. wieder auf Objekt ausrichten!")
while get_key() != "esc":
    r2 = drone.get_front_range()
    print(r2)
    sleep(1)
print("r1:",r1,"r2",r2)
print("Experiment speichern j/n?")
if get_key(1) == "j":
    add_experiment_data(v_omega,v_seite,komp_zentrifugal,r1,r2,delta,dau
    print("gespeichert!")
drone.close()
```

3. Nebenstehend der Code des „selfie_archive_funcs“-Moduls.

Zunächst wird geprüft, ob bereits die Nspire-Variablen von einem vorherigen Experiment vorhanden sind. Falls ja, wird geprüft, ob die Nspire-Listen leer sind. In beiden Fällen werden die temporären Python-Listen neu initialisiert. Falls jedoch schon Daten von vorherigen Experimenten vorhanden sind, werden die aktuellen Daten in die temporären Python-Listen hinzugefügt und anschließend der neue Datensatz in die Nspire-Listen abgespeichert.

selfie_archive_funcs.py 1/40

```
from ti_system import store_list,recall_list

def add_experiment_data(omega,v,komp,r1,r2,delta,dauer):
    try:
        n_exp = recall_list("n")
        dummy = n_exp[-1]
        omega_exp = recall_list("omega")
        v_exp = recall_list("v")
        komp_exp = recall_list("z")
        r1_exp = recall_list("r1")
        r2_exp = recall_list("r2")
        dx_exp = recall_list("winkel")
        d_exp = recall_list("dauer")
    except:
        n_exp = [0]
        omega_exp = []
        v_exp = []
        komp_exp = []
        r1_exp = []
        r2_exp = []
        d_exp =[]
        dx_exp =[]
    if n_exp[-1] > 0:
        n_exp.append(n_exp[-1]+1)
    omega_exp.append(omega)
    v_exp.append(v)
    komp_exp.append(komp)
    r1_exp.append(r1)
    r2_exp.append(r2)
    dx_exp.append(delta)
    d_exp.append(dauer)
    store_list("n",n_exp)
    store_list("omega",omega_exp)
    store_list("v",v_exp)
    store_list("z",komp_exp)
```

© Texas Instruments EMEA Sales GmbH 2026

	store_list("r1",r1_exp) store_list("r2",r2_exp) store_list("winkel",dx_exp) store_list("dauer",d_exp)

v. Reflektion, Gelerntes

Sie haben Ihr physikalisches und mathematisches Verständnis für die komplexen Flugmanöver einer Drohne vertieft. Experimentell haben Sie mit dem 360° Selfie ein automatisches Flugmanöver kennengelernt, das in vielen kommerziellen Kamera-Drohnen als autonomes Flugmanöver zur Verfügung steht. Sie haben verstanden, wie sich Dreh- und Bewegungs-Parameter auf die Flugbahn der Drohne auswirken und wie man sie einstellt, um eine kreisförmige Flugbewegung zu erhalten. Sie haben gelernt, wie man sich experimentell an ein gewünschtes Ziel annähert. Nicht zuletzt ist das in dieser Übung verwendete Modul zum Abspeichern experimentell gefundener Messdaten vielleicht auch hilfreich für andere Experimentier-Versuche in Ihrem Schulalltag.

vi. Anregungen und Erweiterungen

Ändern Sie die Flugmission, dass die CoDrone den virtuellen Kreismittelpunkt mit ihrer Nase in Flugrichtung umrundet. Programmieren Sie eine weitere Flugmission, bei der die Drohne mit der Nase entgegengesetzt des Kreismittelpunkts fliegt (360° Panorama-Flug). Erweitern Sie Ihr Programm um die dritte Dimension und lassen die Drohne einen spiralförmigen Steigflug beim Umrunden des Objekts machen.

Diskutieren Sie ihre Experimente und die gewonnen Erkenntnisse mit Ihren Mathematik- und Physiklehrern. Versuchen Sie, eine mathematische Beziehung zwischen den Bewegungs-Größen aufzustellen. Beweisen Sie die Richtigkeit Ihrer Formel in einem weiteren Flugexperiment. Erstellen Sie ein Python Programm, bei dem nur der Kreisradius und die Winkelgeschwindigkeit vorgegeben wird und alle anderen Parameter vom Programm berechnet werden. Hinweis: Laut Datenblatt wiegt die CoDrone flugfertig (inklusive Akku) 58 g.

© Texas Instruments EMEA Sales GmbH 2026

5. Informationen für Lehrkräfte und Workshopleiter

5.1 Allgemeines

Die CoDrone Indoor-Drohne eignet sich sehr gut zur praxisorientierten Ergänzung des Mathematik-, Physik- oder Informatik-Unterrichts in allen Jahrgangsstufen der Sekundarstufe I und II. Einige Schüler und Schülerinnen haben vielleicht schon Erfahrung mit der Steuerung von Drohnen im Consumer-Bereich zum Erstellen von Videos oder mit einer FPV-Renndrohne für Flugakrobatik und Geschicklichkeit. Die wenigsten unter ihnen wissen, dass es auch Lern-Drohnen wie die CoDrone gibt, die man mit einer Programmiersprache steuern kann. Fragen Sie Ihre Schüler und Schülerinnen nach ihren Erfahrungen und Wissen über Drohnen und ihren Einsatzbereich und lassen Sie sie eine erste Stunde gestalten und die allgemeinen Grundlagen für den Workshop selbst erarbeiten.

Im Workshop wird das TI-Nspire™ CX II Einführungshandbuch[6] mehrfach erwähnt. Dieses Dokument enthält wichtige Grundlagen zur Inbetriebnahme des TI-Nspire™ CoDrone Systems, eine Python-Funktionsreferenz und Wissenswertes zum Betrieb und zur Fehlersuche. Diese Elemente werden hier als bekannt vorausgesetzt.

Die CoDrone besitzt kein GPS, sondern ermöglicht die Orientierung und das Schweben über einem bestimmten Punkt durch drei wichtige Sensoren, wie rechts im Bild zu sehen. Während die beiden RGB-Sensoren der Farberkennung des Bodens bei gelandeter Drohne dienen, werden der ToF-Sensor und der Bildmuster-Sensor zur Positionsregelung verwendet. ToF-Sensoren senden gepulstes oder moduliertes Infrarotlicht aus (meist per LED oder Laser) und messen die Laufzeit bzw. Phasenverschiebung des reflektierten Lichts, um Entfernungen zu bestimmen.

Dies führt zu respektabler Flug-Genauigkeit der CoDrone im Indoor-Bereich, ist aber für den Outdoor-Betrieb nicht geeignet. Ein strukturierter und gemusterter Boden ist entscheidend für die Positionsgenauigkeit der CoDrone. Sollten Sie erratische Flugbewegungen oder ein Driften der Drohne feststellen, probieren Sie das Flugmanöver über einem anderen Untergrund, bevor Sie Fehler in der Hardware der Drohne vermuten.

[6] TI-Nspire™ CX II – CoDrone EDU Einführungshandbuch
https://education.ti.com/de/product-resources/codrone-edu

5.2 Überblick über die Workshop-Übungen und Zeitbedarf

Hier erhalten Sie einen Überblick über die im Workshop angebotenen Übungen und den ungefähren Zeitbedarf. Der minimale Zeitbedarf bezieht sich auf die reine Durchführung der Kernübung mit erfahrenem Lehrpersonal und technisch versierten Schülergruppen. Der maximale Zeitbedarf umfasst auch den wesentlichen Teil der unter dem Punkt „Anregungen und Erweiterungen“ aufgeführten Zusatzaufgaben.

Übungsnummer	Übung	Thema	Aufbauend auf	Nutzt TI-Nspire™ Lists & Spreadsheet-Fenster	Zeitbedarf (Minuten)
1.1	Einfache Flugmissionen	Kommando-basierte Flüge	--	nein	45-90
1.2		Wegpunkt-basierte Flüge	1.1	ja	45-90
1.3		Bewegungsvektor-basierte Flüge	1.2	ja	90-120
2.1	Messwerterfassung mit der CoDrone	Höhenmessung oder Luftdruckmessung?	1.3	ja	90-120
2.2		Wie lange fliegt die CoDrone?	2.1	ja	90-120
2.3		Interaktive Objekterkennung mit der CoDrone	2.1	nein	90-180
3.1	Komplexe Flugmanöver	Physikalisch-mathematische Grundlagen	--	--	Je nach Jahrgangsstufe
3.2		360° Selfie	1.2	ja	90-270

5.3 Kompatibilität zwischen TI-Nspire™ und Robolink CoDrone EDU Plattform

Einheit	Lektion	Kompatibel mit TI-Nspire™ CX II?	Anmerkungen / Kommentare
0 Einstieg	0.1 Willkommen bei Python	Ja	Einführung in Python.
	0.2 Python für Robolink	N/A	Online-Web-Anwendung.
	0.3 PyCharm	N/A	Nutzung von PyCharm.
1 Anfänger	1.1 Erster Flug	Ja	Erstflug-Übungen.
	1.2 Flugbewegungen: Rollen & Nicken	Ja	Steuerung von Roll & Pitch.
	1.3 Flugbewegungen: Schub & Gieren	Ja	Steuerung von Throttle & Yaw.
	1.4 Variablen 1: Datentypen	Ja	Einführung in Datentypen.
	1.5 Variablen 2: Verwendung	Ja	Variablennutzung.
	1.6 Bedingungen	Ja	If-Statements und Logik.
	1.7 For-Schleifen	Ja	Zählschleifen.
	1.8 While-Schleifen	Ja	Bedingungsschleifen.
2 Fortgeschrittene	2.1 LEDs und Summer	Ja	set_drone_LED() verwendet in TI-Nspire™ nur drei Argumente.
	2.2 Void-Funktionen	Ja	Funktionen ohne Rückgabewert.
	2.3 Return-Funktionen	Ja	Funktionen mit Rückgabewert.
	2.4 Listen	Ja	TI-Nspire™ bietet store_list() zum Speichern in Nspire-Variablen
	2.5 Zufall	Ja	Verwendet nur randint().
	2.6 Timer	Ja	Zeitmessungen
3 Profi	3.1 Sensoren Einführung	Ja	Grundlagen der Sensorik.
	3.2 Batterie	Ja	TI-Nspire™ bricht Programm ab, wenn Ladestand < 50 % ist.
	3.3 Temp & Druck	Ja	Barometrische Messungen.
	3.4 Gyroskop	**Nein**	Keinen direkten Zugriff.
	3.5 Höhe	Ja	Höhenmessung.
	3.6 Front-Sensor (Range)	Ja	avoid_wall() nicht verfügbar.
	3.7 Farbe	Ja	get_colors() liefert Liste.
4 Experte	4.1 Benutzereingaben	Ja	Verwendet get_key() aus dem ti_system Modul.
	4.2 Tastatur-Steuerung	Ja	Steuerung über die Tasten des Taschenrechners.
	4.3 Eigener Smart-Controller	**Nein**	Nicht verfügbar auf TI-Nspire™ CX II.

© Texas Instruments EMEA Sales GmbH 2026

6. Literatur

[1] TI-Nspire™ CX II – CoDrone EDU Einführungshandbuch
https://education.ti.com/de/product-resources/codrone-edu

[2] Luftfahrtbundesamt (LBA) Die neue Drohnen-Verordnung - Ein Überblick über die wichtigsten Regeln (BMDV)
https://www.lba.de/SharedDocs/Downloads/DE/B/B5_UAS/Drohnenflyer_BMVI.html

[3] van Hees, H. (2025). Newton'sche Mechanik. In: Mechanik für das gymnasiale Lehramt. Springer Spektrum, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-69047-5_2

© Texas Instruments EMEA Sales GmbH 2026