Arbeitsblatt TelloSimulator

Vorbereitung 1: Setup TelloSimulator

Schritt 0: Für den TelloSimulator wird ein aktuelles Java JDK mit integriertem JavaFX benötigt. Bitte installiere deshalb den “Full JDK” des LibericaJDK. Auf der Download-Seite https://bell-sw.com/pages/downloads/#/java-15-current kannst du den passenden Liberica JDK 15.0.1+9 für dein Betriebssystem herunterladen.

Schritt 1: Repository in gewünschtes lokales Verzeichnis klonen (z.B. mit Command Prompt / Git Bash / Terminal):

```bash
git clone https://github.com/DieterHolz/TelloSimulator
```

Schritt 2: Ins TelloSimulator Verzeichnis wechseln:

```bash
cd TelloSimulator
```

Schritt 3: Starten des Simulators via Gradle Wrapper:

```bash
gradlew run
```

oder

```bash
./gradlew run
```

Vorbereitung 2: Setup TelloPlaygroundApp

Schritt 1: TelloPlaygroundApp-Code von Teams downloaden (**module09-start.zip**) und in das emoba-Projekt integrieren.

Schritt 2: In TelloPlaygroundApp.kt auf Zeile 15 deine Simulator IP (im TelloSimulator GUI rechts ersichtlich) einsetzen.

```kotlin
val connector = TelloConnector(ip = "YourSimulatorIP",  //real Tello: 192.168.0.12
                               commandPort = 8879,      //real Tello: 8889
                               statePort   = 8890)
```

Schritt 3: TelloPlaygroundApp auf Emulator oder Android Device starten.

Schritt 4: Die virtuelle Drohne im Simulator mit dem Button "Start Drone" einschalten.

Schritt 5: Von der TelloPlaygroundApp aus mit dem Button "Connect" sich mit der virtuellen Drohe verbinden.

Die Drohne sollte nun von der TelloPlaygroundApp aus steuerbar sein.

Wenn der Simulator 20 Sekunden lang kein Command erhält, schliesst er automatisch die Verbindung. Um die Verbindung zurückzusetzen, kann die virtuelle Drohne im Simulator einfach mit "Stop/Start Drone" wieder aus- und eingeschaltet werden.

Wir empfehlen an diesem Punkt kurz die TelloPlaygroundApp und die bereits implementierten Commands zu studieren.

Aufgabe 1: «flip»-Command

Das «flip»-Command wird mit einem Paramter «l» (left), «r» (right), «f» (forward) oder (also z.B. «flip b») (backward) verschickt. Die Drohne führt damit einen Salto in die entsprechende Richtung durch. (also z.B. «flip b» um einen Rückwärts-Salto durchzuführen)

Implementiere in der TelloPlaygroundApp einen Button, um der Drohne das command «flip» inkl. Paramter (z.B. «flip b») zu schicken.

Erhält die Drohne ein «flip»-Command, dann schickt sie nach erfolgreicher Ausführung die Antwort "ok" zurück. Erst nachdem diese Anwort verschickt wurde, werden weitere Commands wieder angenommen. Du solltest also die bereits vorhandene `sendCommandAndWait` -Funktion verwenden.

Als Vorlage kann dir das bereits implementierte «forward»-Command dienen.

Aufgabe 2: «rc»-Command

Das «rc»-Command (rc für Remote Controller) ist wohl das zentralste Command zur Fernsteuerung der Tello-Drohne. Hiermit können auf den vier Kanälen **(links/rechts)**, **(vorwärts/rückwärts)**, **(hoch/runter)** und **(yaw)** die Geschwindigkeiten gesetzt werden. Damit lässt sich die Drohne eine beliebige Richtung im Raum bewegen.

Passe den bereits implementierten left/right-Slider so an, dass man sich damit neu hoch/runter bewegen kann. (Alternativ kannst du auch einen dritten Slider dafür einbauen)

Das «rc»-Command funktioniert zu jeder Zeit und erwartet auch keine Antwort von der Drohne. Deshalb wird bei der Implementation die `fireAndForgetCommand` -Funktion verwendet.

Aufgabe 3: Status auslesen

Die Tello-Drohne verschickt alle 100 ms einen Status mit diversen nützlichen Parameter-Werten der Drohne. Der Status-String wird dabei von der TelloPlaygroundApp über den statusSocket empfangen.

Implementiere ein Textfeld in der TelloPlaygroundApp, um den aktuellen Batterieladestand gemäss Drohnen-Status anzuzeigen.

Ein Beispiel eines State-Strings findest du im Anhang unter Weitere Eigenschaften der Tello-API.

Als Vorlage kann dir der bereits implementierte «height»-Text dienen.

Anhang

Programmierschnittstelle

Dokumentation aller Commands basiernd auf dem Tello SDK 2.0. Dank ausführlichen Tests mit der Tello-Drohne beinhaltet die folgende Auflistung aber detailliertere und vollständigere Beschreibungen zu den einzelnen Commands.

> **Für alle Commands gilt:** Unbekannte oder falsch geschriebene Commands liefern die Antwort «unknown command:» gefolgt vom gesendeten String. Commands sind dabei Case-sensitive, d.h. Gross- und Kleinschreibung muss eingehalten werden.

Table 1. Control Commands

Command	**Beschreibung**	**Mögliche Antworten**	**Simulator**
command	**Enter SDK mode**. Die Drohne ist ab jetzt via commands steuerbar. Ein zweites command zu senden gibt zwar «ok» zurück, hat aber keine weiteren Auswirkungen.	ok / error	[✓]
takeoff	**Auto takeoff**. Startet die Motoren und fliegt 30 cm nach oben. Wird nach dem initialen takeoff ein weiteres takeoff gesendet, wird es ignoriert und die Drohne schickt «error» als Antwort.	ok / error	[✓]
land	**Auto landing**. Fliegt nach unten bis der Boden erreicht ist und stoppt anschliessend die Motoren. Schickt «ok» nach der Landung.	ok / error	[✓]
streamon	**Enable video stream**. Startet die Video-Übertragung.	ok / error	
streamoff	**Disable video stream**. Stoppt die Video-Übertragung.	ok / error	
emergency	**Stops motors immediately**. Stoppt die Motoren, fällt auf den Boden. Sendet weiterhin den Status. Hinweis: Funktioniert zu jeder Zeit.	keine Antwort	[✓]
up x	**Ascend to x cm**. *x = 20-500*. Bewegt sich x cm nach oben. Nachdem die Drohne sich wieder stabilisiert hat wird die Antwort «ok» versendet	ok / error / out of range	[✓]
down x	**Descend to x cm**. *x = 20-500* . Bewegt sich x cm nach unten. Nachdem die Drohne sich wieder stabilisiert hat wird die Antwort «ok» versendet.	ok / error / out of range	[✓]
left x	**Fly left for x cm**. *x = 20-500*. Bewegt sich x cm nach links. Nachdem die Drohne sich wieder stabilisiert hat wird die Antwort «ok» versendet.	ok / error / out of range	[✓]

Command	Beschreibung	Mögliche Antworten	Simulator
right x	**Fly right for x cm**. *x = 20-500*. Bewegt sich x cm nach rechts. Nachdem die Drohne sich wieder stabilisiert hat wird die Antwort «ok» versendet	ok / error / out of range	[✓]
forward x	**Fly forward for x cm**. *x = 20-500*. Bewegt sich x cm nach vorne. Nachdem die Drohne sich wieder stabilisiert hat wird die Antwort «ok» versendet.	ok / error / out of range	[✓]
back x	**Fly backwards for x cm**. *x = 20-500*. Bewegt sich x cm nach hinten. Nachdem die Drohne sich wieder stabilisiert hat wird die Antwort «ok» versendet.	ok / error / out of range	[✓]
cw x	**Rotate x degrees clockwise**. ~~*x = 1-360*~~. Dreht die Drohne um x Grad im Uhrzeigersinn um seine Yaw-Achse. Der Range Check 1-360 Grad ist zwar in der SDK dokumentiert, wird von der Drohne sowie dem Simulator aber nicht angewandt. Auch negative Werte sind möglich.	ok / error	[✓]
ccw x	**Rotate x degrees counterclockwise**. ~~*x = 1-360*~~. Dreht die Drohne um x Grad im Gegenuhrzeigersinn um seine Yaw-Achse. Der Range Check 1-360 Grad ist zwar in der SDK dokumentiert, wird von der Drohne sowie dem Simulator aber nicht angewandt. Auch negative Werte sind möglich.	ok / error	[✓]
flip x	**Flip in x direction**. *x = l | r | f | b*. Führt einen Salto in die angegebene Richtung aus.	ok / error / out of range	[✓]
go x y z speed	**Fly to x y z at speed (cm/s)**. *x = -500-500, y = -500-500, z = -500-500, speed = 10 - 100*. Fliegt zu den entsprechenden Koordinaten (relativ zur aktuellen Drohnenposition). Achsen: x = Drohnenausrichtung, y = Linker Normalvektor der Drohne, z = Aufwärtsvektor der Drohne. Hinweis: x-, y- und z-Werte können nicht gleichzeitig zwischen -20 - 20 eingestellt werden.	ok / error / out of range	[✓]
stop	**Hovers in the air**. Unterbricht die Ausführung eines Commands und stoppt die Drohne an ihrer aktuellen Position. Hinweis: Funktioniert zu jeder Zeit.	ok / forced stop / error	[✓]

Command	Beschreibung	Mögliche Antworten	Simulator
curve x1 y1 z1 x2 y2 z2 speed	**Fly at a curve according to the two given coordinates at speed (cm/s)**. *x1, x2 = -500-500, y1, y2 = -500-500, z1, z2 = -500-500, speed = 10 - 60*. Mit Hilfe der aktuellen Drohnenposition als Punkt (0,0,0) und der beiden gegebenen Punkten (relativ zur aktuellen Drohnenposition) wird ein Kreisbogen im Raum konstruiert. Die Drohne fliegt auf dieser Kurve bis sie am Endpunkt (x2, y2, z2) angelangt ist. Hinweis 1: Wenn x1, y1 und z1 oder x2, y2 und z2 gleichzeitig zwischen -20 und 20 sind, schickt die Drohne die Antwort «out of range». Hinweis 2: Wenn der Bogenradius nicht innerhalb eines Bereichs von 0,5-10 Metern liegt, schickt die Drohne die Antwort «error Radius is too large!».	ok / error / out of range / error Radius is too large!	[✓]
go x y z speed mid	**Fly to the x y z coordinates of the Mission Pad at speed (cm/s)**. *mid = m1-m8, x = -500-500, y = -500-500, z = -500-500, speed = 10 - 100*. Fliegt zu den entsprechenden Koordinaten (relativ zur aktuellen Drohnenposition). Achsen: x = Drohnenausrichtung, y = Linker Normalvektor der Drohne, z = Aufwärtsvektor der Drohne. Hinweis: x-, y- und z-Werte können nicht gleichzeitig zwischen -20 - 20 eingestellt werden.	ok / error / out of range	
curve x1 y1 z1 x2 y2 z2 speed mid	**Fly at a curve according to the two given coordinates of the Mission Pad ID at speed (cm/s)**. *x1, x2 = -500-500, y1, y2 = -500-500, z1, z2 = -500-500, speed = 10 - 60*. Hinweis 1: Wenn x, y und z gleichzeitig zwischen -20 und 20 sind, schickt die Drohne die Antwort «out of range». Hinweis 2: Wenn der Bogenradius nicht innerhalb eines Bereichs von 0,5-10 Metern liegt, schickt die Drohne die Antwort «error Radius is too large!».	ok / error / out of range / error Radius is too large!	

Command	Beschreibung	Mögliche Antworten	Simulator
jump x y z speed yaw mid1 mid2	**Fly to coordinates x, y and z of Mission Pad 1, and recognize coordinates 0, 0, z of Mission Pad 2 and rotate to the yaw value**. *mid = m1-m8, x = -500-500, y = -500-500, z = -500-500, speed = 10 - 100 (cm/s).* Hinweis 1: Wenn x, y und z gleichzeitig zwischen -20 und 20 sind, schickt die Drohne die Antwort «out of range».	ok / error / out of range	

Table 2. Set Commands

Command	Beschreibung	Mögliche Antwort	Simulator
speed x	**Set speed to x cm/s**. *x = 10-100*. Setzt die Speed-Variable der Drohne auf den entsprechenden Wert.	ok / error	[✓]
rc a b c d	**Set remote controller control via four channels**. *a = left/right (-100-100), b = forward/backward (-100-100), c = up/down (-100-100), d = yaw (-100-100)*. Setzt die Bewegungsgeschwindigkeiten in cm/s in die entsprechende Richtung. Die Werte sind unabhängig von der gesetzten speed-Variable auf der Drohne. Hinweis 1: Funktioniert zu jeder Zeit und schickt kein ok. Hinweis 2: Wenn während einer Sekunde nur 20 nach vorne gegeben werden, reicht das meistens noch nicht aus, um die Drohne zu bewegen. Es braucht mind. 30 in eine Richtung während zwei Sekunden, um die Drohne überhaupt aus dem Gleichgewicht zu bringen.	out of range	[✓]
wifi ssid pass	**Set Wi-Fi password**. *ssid = updated Wi-Fi name, pass = updated Wi-Fi password.*	ok / error	[✓]
mon	**Enable mission pad detection (both forward and downward detection)**.	ok / error	
moff	**Disable mission pad detection**.	ok / error	
mdirection x	**Change mission pad detection mode**. *x = 0/1/2, 0 = Enable downward detection only, 1 = Enable forward detection only, 2 = Enable both forward and downward detection.*	ok / error	
ap ssid pass	**Set the Tello to station mode, and connect to a new access point wit the access point's ssid and password**. *ssid = updated Wi-Fi name, pass = updated Wi-Fi password.*	ok / error	

Table 3. Read Commands

Command	Beschreibung	Mögliche Antwort	Simulator
speed?	**Obtain current speed (cm/s)**. Gibt den aktuell gesetzten Wert der speed-Variable zurück.	x = 10-100 z.B. `100.0\r\n`	[✓]
battery?	**Obtain current battery percentage**. Gibt den aktuellen Batterieladestand zurück.	x = 0-100 z.B. `76\r\n`	[✓]
time?	**Obtain current flight time**. Gibt zurück, wie lange die Drohne bereits geflogen ist, seit sie eingeschaltet wurde (in Sekunden). Wenn die Drohne nach der Landung also nicht ausgeschaltet und wieder takeoff gesendet wird, wird die Zeit einfach aufsummiert.	z.B. `24s\r\n`	[✓]
wifi?	**Obtain Wi-Fi SNR**. Gibt das Wi-Fi Signal-to-Noise Ratio zurück. Hardcoded `90` für den Simulator	z.B. `90\r\n`	[✓]
sdk?	**Obtain the Tello SDK version**. Gibt die SDK Version zurück. Z.B. (Tello SDK 2.0) = `20` für den Simulator	z.B. `20\r\n`	[✓]
sn?	**Obtain the Tello serial number**. Gibt die Seriennummer der Drohne zurück.	z.B. `0TQDG3UEDBSP12`	[✓]

Hinweis: Antworten von Read Commands sowie der Status der Drone enthalten teilweise new line characters `\r\n`.

Weitere Eigenschaften der Tello-API

- Es gibt Commands die ein «ok» zurückschicken, aber auch **«Fire and Forget»-Commands** wie z.B. das rc-Command, welches zu jedem Zeitpunkt von der Drohne ausgeführt wird und keine Antwort schickt.
- Commands werden von der Drohne **nicht gequeued**. Das heisst das Client-Programm muss entsprechend programmiert werden, damit es der Drohne das nächste Command zum passenden Zeitpunkt schickt. Kommt ein Command während ein vorhergehendes noch nicht fertig ausgeführt wurde, dann wird die Antwort **«error Not joystick»** geschickt. Ausnahmen hierzu sind die rc-, stop- und emergency-Commands, welche zu jeder Zeit funktionieren.
- Mitgeschickte **Parameter der Commands können nicht weggelassen werden**, müssen also immer vorhanden sein. Sonst wird das Command nicht erkannt.
- Die Tello-Drohne beginnt mit dem Senden des Status erst, nachdem sie **das erste command-Command** erhalten hat.
- Parameter mit **Kommastellen** wie z.B. «forward 35.234234» oder «cw 35.23453» werden von der Drohne ausgeführt.
- Wird nach dem land-Command ein weiteres Control-Command geschickt, welches laufende Motoren benötigt, kommt **«error Motor stop»** als Antwort zurück.
- Mit dem rc-Command gesetzte Werte bleiben **auch nach der Landung noch aktiv**. Eine Landung setzt die rc-Werte also nicht zurück und müssen vor einem erneuten Start idealerweise manuell mit einem erneuten Befehl «rc 0 0 0 0» zurückgesetzt werden.

- Beispielhafter State-String:
  `mid:-2;x:-200;y:-200;z:-200;mpry:-1,-1,-1;pitch:0;roll:0;yaw:0;vgx:0;vgy:0;vgz:0;templ:72;temph:75;tof:80;h:60;bat:65;baro:281.96;time:27;agx:24.00;agy:1.00;agz:-1001.00;\r\n`

Last updated 2020-12-15 11:22:20 +0100