Tello Drone programmeren - Python

Naast DroneBlocks en Scratch geeft de fabrikant Ryze Robotics ook een ander voorbeeld om de Tello Drone te besturen. In het begin van het Tello SDK pdf-document staat een link <https://terra-1-g.djicdn.com/2d4dce68897a46b19fc717f3576b7c6a/Tello%20%E7%BC%96%E7%A8%8B%E7%9B%B8%E5%85%B3/Both/Tello3(1).py> naar Tello3.py of Tello3(1).py bestand.

Dit programma dient als voorbeeld om de Tello te besturen. Het is geschreven in Python, een populaire programmeertaal die geen blokken, maar tekst gebruikt.

Het is niet al te lang, maar toch wat lastig te begrijpen. Zo worden er bijvoorbeeld *Threads* gebruikt. Verder moet je de opdrachten met de hand intypen, wat minder gebruiksvriendelijk is, want je moet deze kennen of opzoeken.

Daarom gaan we na het uitproberen van dit programma ook nog enkele andere eenvoudigere Python-programma's proberen. Deze programma's gebruiken de standaard Python *libraries*. Dat wil zeggen dat je naast het installeren van Python er niets meer bij hoeft te downloaden of installeren.

Ten slotte volgt nog een klein programma om de videostream van de Tello te zien. Daarvoor moet je naast Python nog wel wat extra downloaden, waardoor het iets ingewikkelder is.

We gaan ervan uit dat je Windows gebruikt. Zo wordt er bijvoorbeeld in een voorbeeld hieronder de *msvcrt*-module gebruikt, die werkt alleen onder Windows.

Begin met het installeren van Python. Ga naar de Python website www.python.org en klik in het midden bij *Download Latest* op Python 3.7.3. Scroll naar beneden en download onder de kop *Files* het bestand dat hoort bij je systeem, zoals bijvoorbeeld voor Windows 10 de *Windows x86-64 executable installer*.

Voer dit bestand uit, daarmee installeer je Python 3.7. Controleer eerst of de opties *pip* en de *py launcher* aangevinkt staan en installeer dan Python. Let op, hiervoor zijn administrator-rechten nodig. Je kunt Python 3.7 gewoon naast andere Python-versies installeren. De voorbeeldcode hieronder werkt mogelijk ook met eerdere Python versies, maar dat hebben we niet getest.

Controleer na het installeren of Python goed werkt. Start een *command prompt* (open het Windows *Startmenu* en typ cmd.exe) en type *py*. Je start dan de Python-launcher. Deze zoekt waar python.exe staat en runt deze. De Python-launcher is handig als je meerdere Python-versies op dezelfde computer gebruikt. Je kunt op de *commandline* aangeven welke Python-versie je wilt gebruiken. Ander voordeel is dat python.exe niet in deze *environment* (omgeving) hoeft te staan. Je kunt nu Python interactief gebruiken, maar dat doen we niet. Sluit Python af met *Ctrl+Z* en *Enter*:

```
d:\>py
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> ^Z
```

Ga naar de map met het downloadbestand Tello3(1).py en run dit met *py*:

```
d:\>cd Tello

d:\Tello>py Tello3(1).py


Tello Python3 Demo.

Tello: command takeoff land flip forward back left right
     up down cw ccw speed speed?

end -- quit demo.
```

Als dit de eerste keer dat je Python start na installatie, dan verschijnt er een Windows-beveiligingsmelding, zoals ook in het artikel in PC-Active aangegeven. Sta toe dat Python het *Particuliere en Openbare netwerk* mag benaderen, vink beide opties aan en klik op *Toegang toestaan*. Ook hiervoor heb je administatorrechten nodig.

Zorg voor een werkende wifiverbinding met de Tello Drone, type *command* en druk op enter:

```
command
ok
```

Wanneer er geen *ok* verschijnt dan is mogelijk eerder een ander programma of app gebruikt voor verbinding met de Tello. Zet dan de Tello uit en aan, maak opnieuw (wifi)verbinding en probeer het nog eens.

Zorg er vervolgens voor dat de Tello op een plaats staat zodat deze veilig kan opstijgen. Type dan *takeoff*, de drone stijgt op:

```
takeoff
ok
```

En dan weer landen:

```
land
ok
```

Als je te snel bent met de opdracht *land* dan wordt deze soms genegeerd en krijg je geen *ok* terug. Overigens kan de ontvangst van *ok* soms even duren, gebruik daarom niet te snel het volgende commando.

Soms werkt het opstijgen niet:

```
takeoff
error
battery?
6
```

Dan kun je met *battery?* zien dat de batterij bijna leeg is. Laadt deze dan eerst op of verwissel deze met een volle.

Zo kun je nog veel meer opdrachten uitvoeren, zie ook het Tello SDK pdf-document. De opdrachten (*commands*) komen overeen met de blokken zoals je die eerder hebt gezien in DroneBlocks of Scratch.

Je kunt op deze manier ook de opdracht *streamon* naar de Tello sturen Zo kun je de videostream met camerabeelden gebruiken met VLC, zoals eerder beschreven. Dat is mogelijk handiger dan een *streamon* via Scratch, waar je ook nog Node met Tello.js nodig hebt.

Vergeet niet dat je als eerste een command - opdracht dus - naar de Tello moet sturen.

Stop het programma met *end*:

```
battery?
81

end
...

Exit . . .
```

Met het volgende kleine Python-programma testen we de verbinding en de batterij:

```
import socket

def mainCmd1():
    conn = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    conn.bind(('', 9000))
    conn.connect(('192.168.10.1', 8889))
    conn.settimeout(5.0)

    conn.send(b'command')
    for i in range(5):
        resp = conn.recv(1024)
        print(resp.decode(errors='replace').strip())
        conn.send(b'battery?')

if __name__ == '__main__':
    mainCmd1()
```

Plak de code in een bestand *TelloCmd1.py* en run dit:

```
d:Tello>py TelloCmd1.py ok 74 74 74 74
```

De *import socket* geeft toegang tot de netwerkmodule.

De meeste code staat in de *mainCmd1*-functie. Het is een goede gewoonte code zoveel mogelijk in een functie te zetten, dan kun je de code ook gebruiken in andere delen van je programma en zijn de meeste variabelen die gebruikt worden lokaal, onafhankelijk van de rest.

De *mainCmd*-functie wordt niet meteen uitgevoerd, alleen de definitie in de regels die volgen en bij de functie horen worden gecontroleerd en bewaard. Dit zijn alle regels die aan het begin meer spaties aan het begin hebben staan, die een hogere indentatie hebben.

Pas als Python de regel `if __name__ == '__main__':` tegenkomt wordt de *mainCmd1*-functie aangeroepen. Dit gebeurt alleen als *TelloCmd1.py* direct wordt aangeroepen zoals we hierboven doen. We kunnen later in een programma `import TelloCmd1` doen, dan is `__name__` niet gelijk aan `__main__` en wordt de functie *mainCmd1* niet meteen uitgevoerd.

Eerst openen we binnen de functie met `socket.socket` een socket met `AF_INET` en `SOCK_DGRAM`. Dit is de standaardmanier om een Internet UDP-verbinding te maken. We gebruiken een functie uit de *socket*-module, dat zie je aan *socket* voor de punt.

Vervolgens wordt met *bind* het lokale IP-adres en de poort vastgelegd. De *bind*-functie heeft 1 argument, een *tuple* met adres en poort. Daarom staan er dubbele haakjes, de binnenste maken een *tuple*.

Een lege string `''` als adres geeft aan dat alle IP-adressen van deze computer gebruikt mogen worden. Zo hoeven we niet te achterhalen wat het eigen IP-adres is na verbinding met de Tello. Waarschijnlijk is dit 192.168.10.2, maar het kan ook 192.168.10.4 zijn, dat maakt zo dus niet uit.

Als lokale poort wordt poort 9000 gebruikt. Vrijwel elk poortnummer zou werken, maar het is beter om een vast poortnummer te gebruiken, anders moeten we telkens de Tello uit en aanzetten en opnieuw wifiverbinding maken. De Tello gebruikt na het aanzetten het poortnummer van de app of van het programma dat de eerste keer verbinding heeft gemaakt. We gebruiken 9000 omdat dit ook door het voorbeeld Tello3(1).py wordt gebruikt. Het aanroepen van de *bind*-functie zou zelfs weggelaten kunnen worden, maar dan wordt elke keer een willekeurig poortnummer gebruikt, automatisch bepaald door het operating system.

Met de *connect* aanroep worden het remote IP-adres en de poort vastgelegd, dit zijn het IPv4-adres en de poort van de Tello Drone. Omdat dit om een UDP-verbinding gaat, wordt er niet echt een connectie gemaakt.

Met `settimeout(5.0)` wordt de timeout op 5 seconden gezet. Lukt het niet om verbinding te maken, dan geeft het programma een *exception* (uitzondering). Die vangen we hier niet op, dan stopt het programma met een melding. Dit gebeurt als je het programma probeert te draaien terwijl er geen verbinding is. De standaard (default) timeout is veel hoger, waardoor je lang zou moeten wachten. Je kunt een Python-programma in een *command prompt* (console) meestal geforceerd afsluiten met *Ctrl+C* of *Ctrl+Break*.

Vervolgens stuurt het programma met *send* een *command*-opdracht, dit moet de eerste opdracht zijn.

De lus (loop) `for i in range(5)` voert de volgende regels 5 keer uit. Hierin wacht het programma met *recv* op een resultaat, en bewaart dit in de *resp* variabele. Het resultaat is een bytestring, deze wordt met `decode` in een gewone string, een *str*-type omgezet. Een bytestring bevat bytes, een reguliere string bevat unicode karakters. Soms stuurt de Tello

binaire waardes, die een fout en een *exception* kunnen geven in *decode*. Met `errors='replace'` worden deze waardes zonder *exception* omgezet. Met de *strip*-functie wordt de *linefeed* aan het einde verwijderd. Deze is niet nodig, de *print*-functie schrijft het resultaat *ok* van de *command*-opdracht naar de uitvoer en gaat aan het einde naar een nieuwe regel.

Daarna stuurt het programma *battery?* naar de Tello Drone en begint de lus opnieuw. Het resultaat is deze keer een percentage. Zo wordt dat enkele keren herhaald, totdat de loop enkele keren is doorlopen

Zodra de Tello Drone de opdracht *command* ontvangt stuurt deze naast een antwoord ook regelmatig een regel met de Tello *state* (status) naar poort 8890.

Poort 8890 wordt niet gebruikt door TelloCmd1.py, deze informatie kan daarom met een ander programma worden opgevangen. Als een programma poort 8890 gebruikt voor de ontvangst van de resultaten, dan komen op diezelfde poort ook de statusberichten binnen. Dit kan soms handig zijn, soms juist niet.

Met het volgende programma *TelloState.py* kunnen we deze regels ontvangen:

```
import socket

def mainState():
    localCmdIpPort = '', 8890
    conn = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    conn.settimeout(1.0)
    conn.bind(localCmdIpPort)

    count = 0
    while True:
        resp = conn.recv(1024)
        print(count, resp.decode(errors='replace').strip())
        count += 1

if __name__ == '__main__':
    mainState()
```

Dit programma opent een *UDP-socket*. In plaats van de *tuple* rechtstreeks door te geven aan *bind*, wordt eerst de lokale variabele *localCmdIpPort* gevuld en later gebruikt. Dit programma ontvangt alleen maar, daarom is een *connect* niet nodig.

De lus blijft vanwege de *while True* onbeperkt lopen, totdat er een fout, een *exception*, optreedt. Dit gebeurt bijvoorbeeld als de verbinding wegvalt, de *recv*-functie geeft dan een *timeout exception* die niet wordt opgevangen.

Zodra je verbinding hebt met de Tello Drone en er via een ander programma zoals TelloCmd1.py een command-opdracht is gestuurd kun je TelloState.py gebruiken:

```
d:\Tello>py TelloState.py
0 pitch:-1;roll:6;yaw:0;vgx:0;vgy:0;vgz:0;templ:58;temph:61;tof:10;h:0
;bat:70;
baro:-77.62;time:0;agx:-14.00;agy:-122.00;agz:-992.00;
```

```
1 pitch:-1;roll:6;yaw:0;vgx:0;vgy:0;vgz:0;templ:58;temph:61;tof:10;h:0
;bat:70;
baro:-77.86;time:0;agx:-14.00;agy:-121.00;agz:-994.00;
2 pitch:-1;roll:6;yaw:0;vgx:0;vgy:0;vgz:0;templ:58;temph:61;tof:10;h:0
;bat:70;
baro:-77.52;time:0;agx:-15.00;agy:-124.00;agz:-993.00;
...
```

Aan het einde van het Tello SDK pdf-document wordt de Tello State nader beschreven.

Als er geen verbinding is of deze valt na een tijd weg, dan stopt het programma met een *socket.timeout exception*:

```
d:\Tello>py TelloState.py
Traceback (most recent call last):
  File "TelloState.py", line 16, in <module>
    mainState()
  File "TelloState.py", line 11, in mainState
    resp = conn.recv(1024)
socket.timeout: timed out
```

Het volgende programma *TelloCmd2.py* is uitgebreider, hiermee kun je verschillende opdrachten naar de Tello sturen:

```
import msvcrt  # Windows only
import socket

def mainCmd2():
    print('Tello cmd demo v0.1 - july 2019')
    cmds = { ' ' : 'land',      'b' : 'back 20',    'c' : 'cw 15',
             'C' : 'ccw 15',    'd' : 'down 20',    'e' : 'emergency',
             'E' : 'emergency', 'f' : 'forward 20', 'g' : 'flip l',
             'h' : 'height?',   'i' : 'temp?',      'j' : 'baro?',
             'k' : 'attitude?', 'l' : 'left 20',    'n' : 'acceleratio
n?',
             'o' : 'tof?',      'r' : 'right 20',   's' : 'streamon',
             'S' : 'streamoff', 't' : 'takeoff',    'T' : 'land',
             'u' : 'up 20',     'w' : 'wifi?',      'y' : 'battery?' }
    help = ('Press z for help, Space to land, e=emergency stop, x=exit
, ' +
        ', '.join(key + '=' + cmd for key, cmd in cmds.items()))
    print(help)
    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as conn:
        conn.settimeout(0.1)
        conn.bind(('', 9000))
        conn.connect(('192.168.10.1', 8889))
        conn.send(b'command')
        while True:
            try:
                resp = conn.recv(1024)
                print(resp.decode(errors='replace').strip())
            except socket.timeout:
                pass
            if msvcrt.kbhit():
                key = msvcrt.getwch()
                if key == 'x':
                    break
```

```
                cmd = cmds.get(key)
                if cmd == None:
                    print(help)
                else:
                    print(cmd)
                    conn.send(cmd.encode())

if __name__ == '__main__':
    mainCmd2()
```

Dit programma gebruikt een *dictionary* (*dict*) voor toetsen met bijbehorende opdrachten. Er is een help-mogelijkheid, zodra een toets wordt gebruikt die niet in de *dict* staat dan wordt de helptekst geprint.

De *socket.socket* wordt geopend met een *with*-statement, dan wordt de socket automatisch gesloten, zelfs bij een exception. Om de recv-functie staat een *try* en *except* om een timeout op te vangen. Het programma wordt dan niet gestopt, `pass` is code die niets doet, er wordt geen output geprint als er niets wordt ontvangen.

Het Tellostate2.py programma stuurt net als TelloState1.py zelf eerst een command-opdracht zodat je dat niet zelf hoeft te doen.

Als er een toets is ingedrukt dan geeft de *kbhit* een waarde *True* en wordt de bijbehorende waarde opgehaald met `getwch`. Met *x* verlaat je het programma, anders wordt de waarde opgezocht in de *dict*. Komt deze niet voor, dan wordt de helptekst geprint, anders wordt de bijbehorende opdracht geprint en naar de Tello gestuurd. Dan wordt de *while-lus* weer doorlopen.

Probeer het TelloCmd2 programma, bij voorkeur eerst met enkele toetsen waarbij vliegen nog niet nodig is:

```
d:\Tello>py TelloCmd2.py
Tello cmd demo v0.1 - july 2019
Press z for help, Space to land, e=emergency stop, x=exit,  =land, b=b
ack 20, c=cw 15,
C=ccw 15, d=down 20, e=emergency, E=emergency, f=forward 20, g=flip l,
h=height?,
i=temp?, j=baro?, k=attitude?, l=left 20, n=acceleration?, o=tof?, r=r
ight 20,
s=streamon, S=streamoff, t=takeoff, T=land, u=up 20, w=wifi?, y=batter
y?
ok
tof?
100mm
streamon
ok
battery?
66
```

Gebruik *x* om het programma te stoppen, of doe dit met *Ctrl+C* of *Ctrl+Break*.

Je kunt met TelloCmd2.py ook met *s* een *streamon*-opdracht naar de Tello versturen. Daarna kun je zoals eerder aangegeven met VLC versie 2 de videostream bekijken.

Met het volgende programma TelloVideoFile.py kun je de videostream opslaan als bestand:

```
import socket
import time

def mainVideoFile():
    localCmdIpPort = '', 11111
    conn = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    conn.settimeout(5.0)
    conn.bind(localCmdIpPort)

    sizeMax = 0
    with open('TelloVideo.h264', 'wb') as videoFile:
        while True:
            fileSize = videoFile.tell()
            print(fileSize, end = '\r')
            if fileSize > 4000000:
                break
            data = conn.recv(2048)
            videoFile.write(data)
            sizeMax = max(sizeMax, len(data))
    print('Maximum packet size', sizeMax)

if __name__ == '__main__':
    mainVideoFile()
```

Run dit programma en wacht even, zodra TelloVideo.h264 4 megabytes bevat stopt het vanzelf:

```
d:\Tello>py TelloVideoFile.py
Maximum packet size 1460
```

Er wordt bijgehouden wat de maximale pakket grootte is, met 1460 ligt dit boven de MTU-limiet die wordt gebruikt voor UDP bij VLC versie 3.

Je kunt het opgeslagen videobestand bekijken in VLC met de opdracht `vlc.exe --demux=h264 TelloVideo.h264`. Dit werkt ook met VLC versie 3, want er wordt een bestand gebruikt en dan is er geen MTU probleem. Wel is dan nog steeds de optie `--demux=h264` nodig.

Ten slotte volgt een programma om de videostreaam van de Tello Drone weer te geven. Dit kan niet met de standaard Python-modules. Hiervoor moet een extra module geïnstalleerd worden. Dit is daardoor wat lastiger dan de eerdere relatief eenvoudige Python-programma's.

We maken hiervoor een *virtual environment* (*venv*):

```
d:\Tello>py -m venv TelVid
```

Dit duurt even, er wordt een nieuwe map *TelVid* gemaakt met daarin een soort van zelfstandige Python-installatie. Hierin kun je modules installeren onafhankelijk van je normale Python-installatie. Je hebt bovendien geen administrator-rechten nodig.

Activeer de nieuwe *virtual environment*, zodat de daarin geïnstalleerde modules worden gebruikt:

```
d:\Tello>TelVid\Scripts\activate

(TelVid) d:\Tello>
```

Als je halverwege stopt en weer verder wilt gaan, hoef je niet opnieuw de virtual environment te installeren. Run opnieuw het *activate*-script zoals hierboven en je kunt verder waar je bent gebleven.

Gebruik *pip* om de OpenCV interface voor Python te installeren:

```
pip install opencv-python
Collecting opencv-python
..
Installing collected packages: numpy, opencv-python
Successfully installed numpy-1.16.4 opencv-python-4.1.0.25
You are using pip version 19.0.3, however version 19.1.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade
pip' command.
```

Update pip is niet echt nodig maar kan wel zoals aangegeven:

```
python -m pip install --upgrade pip
Collecting pip
..
Successfully installed pip-19.1.1
```

Met het volgende TelloVideo.py programma kun je de videostream van de Tello bekijken:

```
import cv2

def MainVideo():
    print('Press x to exit')

    capt = cv2.VideoCapture('udp://0.0.0.0:11111')

    while True:
        result, frame = capt.read()
        cv2.imshow('Tello video', frame)

        key = cv2.waitKey(1) & 0xff
        if key == ord('x'):
            break
    capt.release()
    cv2.destroyAllWindows()

if __name__ == '__main__':
    MainVideo()
```

Run het programma:

```
(TelVid) d:\Tello>py TelloVideo.py
Press x to exit
[h264 @ 0000021eb3719b00] non-existing PPS 0 referenced
[h264 @ 0000021eb3719b00] non-existing PPS 0 referenced
[h264 @ 0000021eb3719b00] decode_slice_header error
..
```

Na enkele seconden verschijnt er een scherm met het videobeeld van de Tello camera.

Je kunt dit programma uitbreiden voor aanvullende effecten. Voeg bijvoorbeeld deze regel toe tussen de regel met *capt.read* en de regel met *cv2.imshow*:

```
frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

Daarmee krijg je een zwart-wit videobeeld.

Zie verder https://docs.opencv.org/3.0.0/dd/d43/tutorial_py_video_display.html hoe je OpenCV in Python kunt gebruiken.

Aanvullende informatie

Er is nog veel meer te vinden over het programmeren van de Tello op internet.

DroneBlocks, Scratch en Python gebruiken de Tello opdrachten (commands) zoals beschreven door de fabrikant in het *Tello SDK* pdf document. Met de standaard Tello app voor Android en iOS zijn meer mogelijkheden beschikbaar, zoals foto's maken of de firmware updaten.

Door te kijken wat er over het Wifi netwerk gaat bij gebruik van deze app is ontdekt dat er naast de Tello SDK tekst interface ook nog een andere binaire interface bestaat. Deze interface is voor een groot deel achterhaald en gedocumenteerd op https://tellopilots.com/wiki/protocol/.

Er zijn al enkele programmeurs bezig geweest met een verbeterde interface naar de Tello, in verschillende programmeertalen. Zie bijvoorbeeld https://tellopilots.com/wiki/development/ of https://github.com/topics/tello.

Daarbij hoort ook https://github.com/dji-sdk/Tello-Python van DJI, de bekende drone fabrikant die onderdelen voor de Tello drone aan Ryze Robotics heeft aangeleverd. Dit project gebruikt nog Python 2.7 dat vanaf 1 januari 2020 niet meer gesupport wordt, zie https://pythonclock.org.

Voor meer informatie over Scratch zie ook de aanvullende informatie bij het seizoensthema Robotica, Drones en Programmeren op www.hcc.nl/robotica-drones-programmeren. Of bekijk de links over Scratch op https://programmeren.hcc.nl/artikelen/online-info-scratch.html.

Voor meer informatie over Python en andere programmeertalen zie de online boekenpagina van HCC!programmeren op https://programmeren.hcc.nl/artikelen/online-boeken.html.

Voor meer informatie over drones, de Tello, andere drones, het bouwen en het vliegen kun je terecht bij HCC!drones https://drones.hcc.nl.