Hackaton 2022

Cheat sheets

Tello connectivity

Connectivity

The tello drone is by default configured with AP mode and controlled connecting directly to the drone's own wifi AP.'

The problem with this configuration is that the computer controlling the drone is not connected to the internet.

The drones used in the hackathon are therefore configured as client mode connected to a Trifork 5G router. Each drone has a fixed IP address and Mac address label on them.

Full reset of the drone will make the drone return to AP mode.

Changing to client mode

Sending the following UDP commands on port 8889 (default) to the drone will put the drone back in client mode. Hint... use the free PacketSender for Mac/Windows

COMMAND

```
Command

ap Hackaton2022 Hackaton2022
```

Tello connectivity

Team	Mac Address	IP address	Default SSID
1	34:d2:62:f2:52:3e	192.168.1.21	TELLO-F2523E
2	34:d2:62:f2:54:18	192.168.1.22	TELLO-F25418
3	34:d2:62:f2:50:f3	192.168.1.23	TELLO-F250F3
4	34:d2:62:f2:52:8f	192.168.1.24	TELLO-F2528F
5	34:d2:62:f2:4c:c9	192.168.1.25	TELLO-F24CC9
6	34:d2:62:f2:51:0e	192.168.1.26	TELLO-F2510E

Tello Flying (Simple)

Flying the drone

The tello drone comes with different options for SDK's for Python, Swift etc. and can also be controlled directly using UDP commands. The examples here are simple Python scripts for simple flying commands.

```
from djitellopy import Tello

tello = Tello()

tello.connect()

tello.takeoff()

tello.move_left(100)

tello.move_right(100)

tello.land()
```

Tello Flying (Swarm Simple)

Flying the drone in swarm

Multiple drone can be controlled together using swarm mode.

```
from djitellopy import TelloSwarm

swarm = TelloSwarm.fromIps([

        "<IP ADDRESS 1>",

        "<IP ADDRESS 2>"])

swarm.connect()

for tello in swarm:

     print(str(tello.get_current_state()))

swarm.takeoff() # take off

swarm.go_xyz_speed(0, 50, 0, 20) # move left

swarm.go_xyz_speed(0, -50, 0, 20) # move right

swarm.land() # landing drones
```

Tello Flying (Mission Pads)

Flying the drone

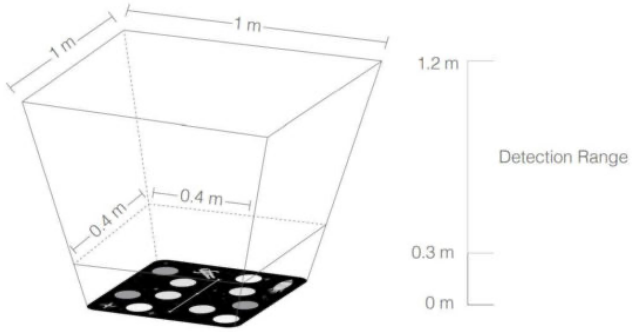
The tello drone comes with a set of mission pads. Drones can be controlled left, right etc. by manually controlling how long they fly in a direction.

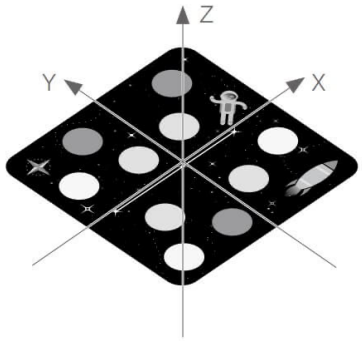
Another and more precise option is using mission pad commands and let the drone find pads. For more information see the link for Mission Pad guide. The drone can be controlled from different API's like Python, Swift etc.

```
from djitellopy import TelloSwarm

speed = 10

swarm = TelloSwarm.fromIps(["192.168.1.180" ])

swarm.connect()

for tello in swarm:

     print("Battery level: " + str(tello.get_battery()) + "%")

     tello.enable_mission_pads()

     tello.streamoff()

     tello.streamon()

swarm.takeoff()

swarm.go_xyz_speed_mid(0, 0, 20, speed, 1) # Align with pad 1 in 40cm height

swarm.go_xyz_speed_mid(0, 0, 120, speed, 1) # Move up to 120cm above pad 2

swarm.go_xyz_speed_mid(0, 0, 20, speed, 1) # Align above pad 1 in 20cm height

swarm.go_xyz_speed_mid(0, 0, 120, speed, 1) # Move up to 120cm above pad 2

swarm.land() # landing drones
```

Tello Flying (Mission Pads)

MISSION PADS





Tello Video

Streaming video

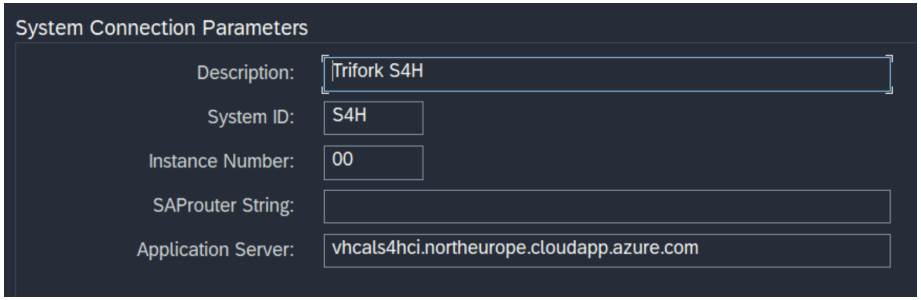
The tello drone has a camera that can be used for both pictures and video streams

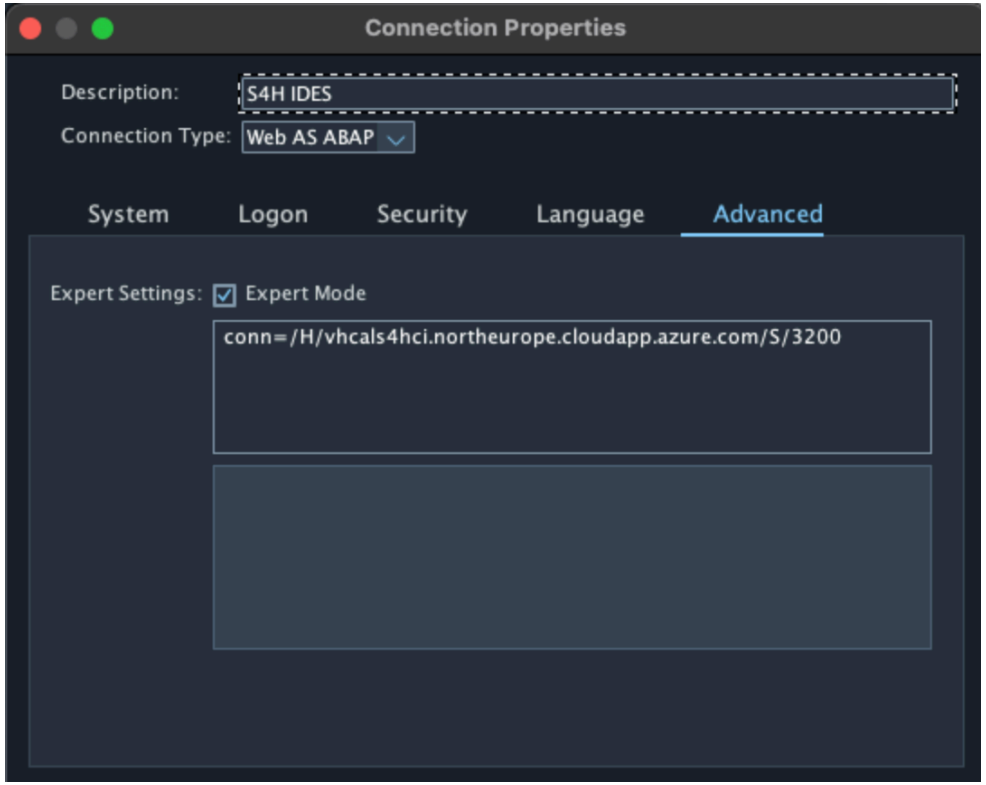
Below an example on a command that can receive and show video stream from the drone using the ip address

FFPLAY UDP://192.168.1.2<TEAM NUMBER>:11111

```
import cv2
from pyzbar.pyzbar import decode

udpStream = 'udp://192.168.10.1:11111' # Connected to the drone default IP Address when in AP mode
# udpStream = 'udp://<IP ADDRESS>>:11111' # Connected to Trifork Hackaton WiFi
# udpStream = 0 # Connected to computers first webcam device

cap = cv2.VideoCapture(udpStream)
while True:
    ret, frame = cap.read()
    # do your processing here
    img = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    scale_percent = 80  # percent of original size
    width = int(img.shape[1] * scale_percent / 100)
    height = int(img.shape[0] * scale_percent / 100)
    dim = (width, height)

    img = cv2.resize(img, dim, interpolation=cv2.INTER_AREA) # resize image
    for barcode in decode(img):
        (x, y, w, h) = barcode.rect
        cv2.rectangle(img, (x, y), (x + w, y + h), (0, 0, 255), 2)
        barcodeData = barcode.data.decode('utf-8')
        barcodeType = barcode.type
        text = "{} ({})".format(barcodeData, barcodeType)
        cv2.putText(img, text, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
        print("[INFO] Found {} barcode: {}".format(barcodeType, barcodeData))
    cv2.imshow("preview", img)
   # Waits for a user input to quit the application
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release() # When everything done, release the capture
cv2.destroyAllWindows()
```

SAP Login and connectivity

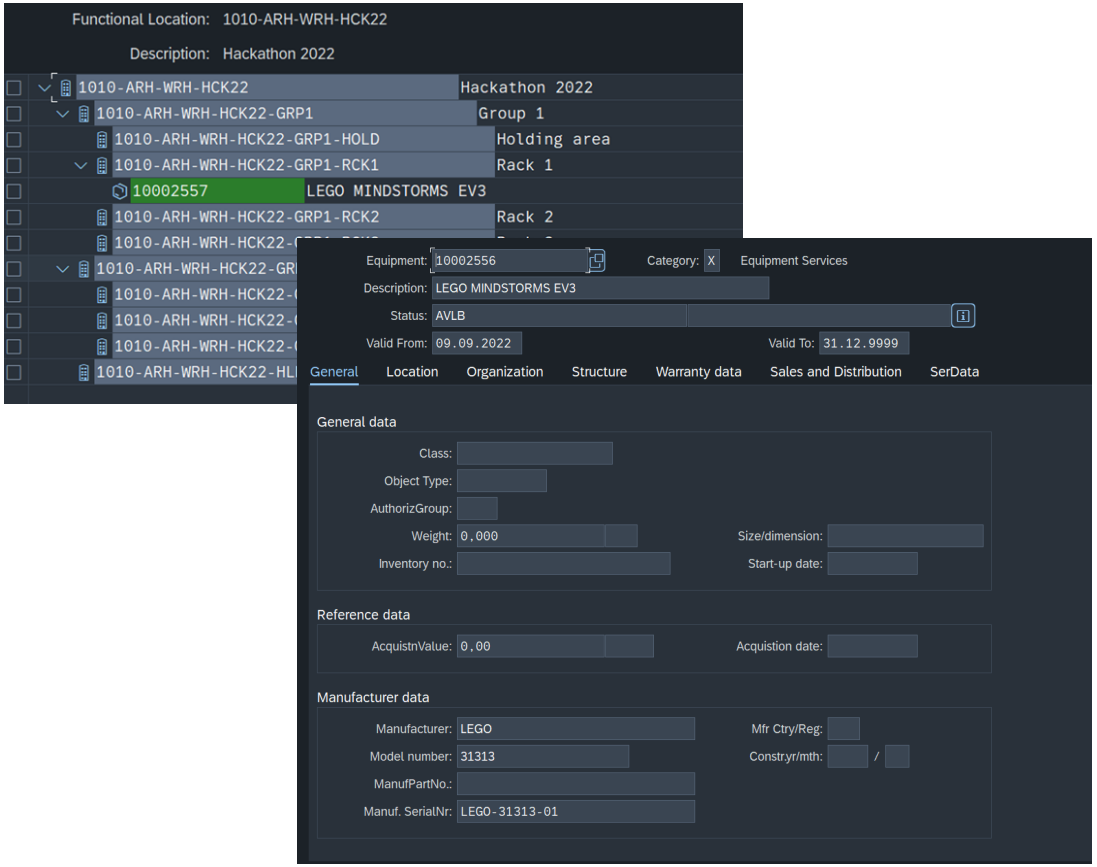




The SAP product and location setup

Each product will refer to an equipment record in SAP and each location will be a functional location in a functional location structure. So the aim of the game is to install the correct equipment (product) into the correct functional location node (shelf/ rack) based on what the drone tells you about which barcode is in which location QR code.

Each equipment will have a material assigned and multiple equipment will have the same material number. So for the product count, you will count how many of each material to we have in a specific location and show in a web-type-portal-view.



SAP Product/Equipment Barcodes (Manufacturer serial number)

Equipment	Material	Barcode	Team 1*	Team 2*	Team 3*	Team 4*	Team 5*
MINDSTORMS EV3	AVC_RBT_ROBOT	LEGO-31313-01	10002557	10002556	10002599	10002600	10002601
MINDSTORMS EV3	AVC_RBT_ROBOT	LEGO-31313-02	10002558	10002559	10002602	10002603	10002604
INVENTOR A1	AVC_RBT_BUNDLE	LEGO-51515-A1	10002560	10002561	10002605	10002606	10002607
INVENTOR A2	AVC_RBT_BUNDLE	LEGO-51515-A2	10002562	10002563	10002608	10002609	10002610
Creator Bot Pack	AVC_RBT_ROBOT2	LEGO-17101-01	10002564	10002565	10002566	10002567	10002568
Creator Bot Pack 2	AVC_RBT_ROBOT2	LEGO-17101-02	10002569	10002570	10002571	10002572	10002573
Creator Bot Pack 3	AVC_RBT_ROBOT2	LEGO-17101-03	10002574	10002575	10002576	10002577	10002578
Creator Bot Pack 4	AVC_RBT_ROBOT2	LEGO-17101-04	10002579	10002580	10002581	10002582	10002583
Controller	AVC_RBT_CNTRL_UNIT	LEGO-18954-01	10002584	10002585	10002586	10002587	10002588
Controller addon	AVC_RBT_CONTROL	LEGO-18988-01	10002589	10002590	10002591	10002592	10002593
Controller addon 2	AVC_RBT_CONTROL	LEGO-18988-02	10002594	10002595	10002596	10002597	10002598

** The SAP **Location/AssetLocation** field will be used to differentiate equipment records per team so ensure you are always filtering for e.g. AssetLocation eq 'TEAM1' when retrieving your equipment records from SAP*

SAP Location / FLOCs QR codes (Manufacturer serial number)

Description	QR code	Team 1*	Team 2*	Team 3*	Team 4*	Team 5*
Row 1 Column 1	HACK22-ROW1-COL1	HCK22-GRP1-R1C1	HCK22-GRP2-R1C1	HCK22-GRP3-R1C1	HCK22-GRP4-R1C1	HCK22-GRP5-R1C1
Row 1 Column 2	HACK22-ROW1-COL2	HCK22-GRP1-R1C2	HCK22-GRP2-R1C2	HCK22-GRP3-R1C2	HCK22-GRP4-R1C2	HCK22-GRP5-R1C2
Row 1 Column 3	HACK22-ROW1-COL3	HCK22-GRP1-R1C3	HCK22-GRP2-R1C3	HCK22-GRP3-R1C3	HCK22-GRP4-R1C3	HCK22-GRP5-R1C3
Row 2 Column 1	HACK22-ROW2-COL1	HCK22-GRP1-R2C3	HCK22-GRP2-R2C3	HCK22-GRP3-R2C3	HCK22-GRP4-R2C3	HCK22-GRP5-R2C3
Row 2 Column 2	HACK22-ROW2-COL2	HCK22-GRP1-R2C3	HCK22-GRP2-R2C3	HCK22-GRP3-R2C3	HCK22-GRP4-R2C3	HCK22-GRP5-R2C3
Row 2 Column 3	HACK22-ROW2-COL3	HCK22-GRP1-R2C3	HCK22-GRP2-R2C3	HCK22-GRP3-R2C3	HCK22-GRP4-R2C3	HCK22-GRP5-R2C3

** Each functional location listed above is prepended with base '1010-ARH-WRH-'.*

** Although the FLoc name itself indicates the Team number, you can also use the Location/AssetLocation field again to filter out FLocs for your specific team only e.g. AssetLocation eq 'TEAM1'*

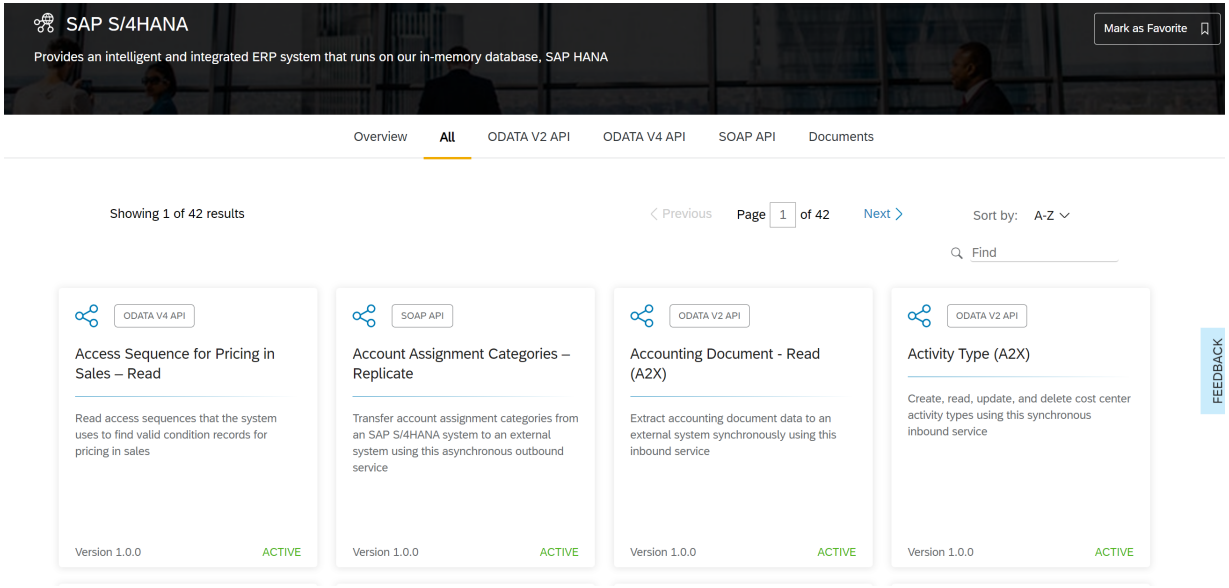
SAP oData / APIs

Remember S/4 Hana provides free oData services that can be easily exposed (if not already). Make use of these in order to make your development process easier and not have to handle building oData or Restful RFCs from scratch.

https://api.sap.com/package/S4HANAOPAPI/all

The API can be pipped through a CAP project by following this CAP guide: https://cap.cloud.sap/docs/guides/using-services#external-service-api

For a working example of this, see repo: https://dev.azure.com/invokersdev/TSE%20Internal/_git/hack22-s4-apis



Links

TELLO

https://www.ryzerobotics.com/tello-edu

https://bestow.info/hacking-the-tello-drone/

https://www.ryzerobotics.com/tello-edu/downloads

https://dl-cdn.ryzerobotics.com/downloads/Tello/Tello%20SDK%202.0%20User%20Guide.pdf

https://dl-cdn.ryzerobotics.com/downloads/Tello/Tello%20Mission%20Pad%20User%20Guide.pdf

https://github.com/dji-sdk/Tello-Python

https://www.youtube.com/watch?v=kcXN7CYgQ0g

SAP

https://cap.cloud.sap/docs/about/

Other tips

Your local firewall software might prevent you from receiving video stream from the drone

See if your drone is alive: ping 192.168.1.2x where x is your team number

Prerequisites

Core Developers

Packet Sender for Mac

Python tools eg. PyCharm or similar, OpenCV, pyzbar

JAVA tools

Etc...

Mobility Developers

Packet Sender for Mac

SAP Developers

xyz

Sample barcodes