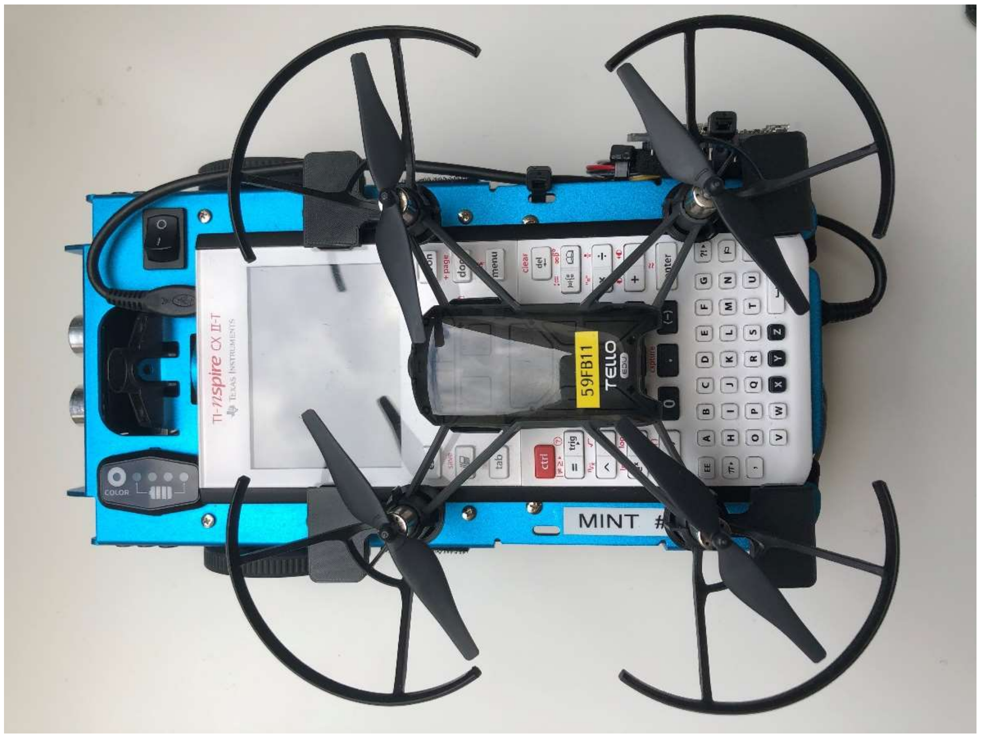
Perseverance & Ingenuity – Control a Tello drone with TI-Innovator

Hans-Martin Hilbig



T³ DEUTSCHLAND

Teachers Teaching with Technology™

Introduction

Year 2021 – NASA brings a rover and a drone to Mars, for the first time ever. What would be more exciting than to bring this technological challenge to the STEM classroom? Let the students figure out how to maneuver the TI-Rover (aka Perseverance) with the Tello (aka Ingenuity) Drone onboard thru a parcours to a place where Ingenuity should take-off, hover, fly over areas impossible to reach with Rover and return to a place where it is safe to land. NASA's Mars mission has inspired me to create Python and Arduino code for the TI-Innovator system to model a rover-drone interaction controlled by a Nspire CX handheld or a Nspire desktop along with a TI-Hub, an ESP8266 NodeMCU and a Tello or Tello EDU drone.

Before you start your Tello drone, please read Ryzerobotics safety notes available on their website [8]!

Hardware overview

Here is a list of hardware components needed to operate the Tello Arduino and Python modules (see links to sources in appendix):

- Computer (PC/Mac) or a TI-Nspire CX/CXII handheld
- TI-Innovator Hub
- ESP8266 NodeMCU WiFi module [1]
- 4-pin Grove-to-female plug adaptor cable [2]
- Ryzerobotics Tello or Tello EDU drone [3]
- 3D-printable Tello mounting pads to slip onto the TI-Rover handheld clamps [4]

The NodeMCU module acts as the interface between the TI-Innovator system and the Tello drone. TI-Hub to NodeMCU communication is done via an UART interface and NodeMCU to Tello communication is done via WiFi. Connect the NodeMCU module to OUT1 Grove plug of a TI-Hub like shown in figure 1. The black (GND) cable connects to the GND pin of the NodeMCU, the red (Vdd) cable to the 3V3 pin, the white cable to the TX pin and the yellow cable to the RX pin of the NodeMCU. All other pins of NodeMCU remain open. **When you connect the NodeMCU module to TI-Hub, make sure you first disconnected the NodeMCU's USB plug. Powering NodeMCU from both, the TI-Hub and the USB port might damage electronics!**

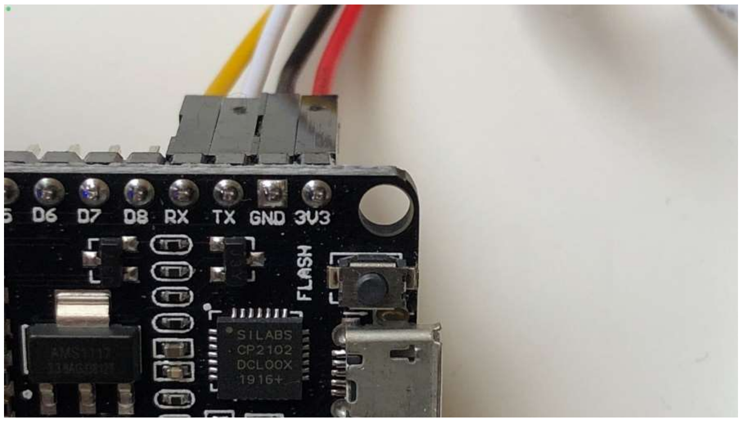


Figure 1 NodeMCU wiring

If you like to use your Tello drone with a TI-Rover, you may print a set of four mounting pads [4] on a 3D-printer. The mounting pads are easy to slide-on / slide-off on the four TI-Rover clamps that secure the handheld onto Rover. See figures 2-4 for details.

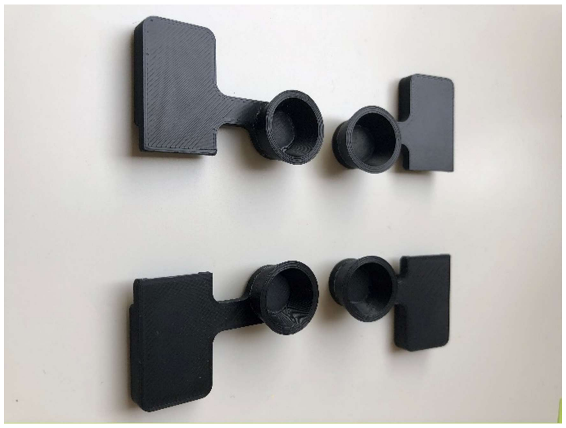


Figure 2 mounting pads (assembled)

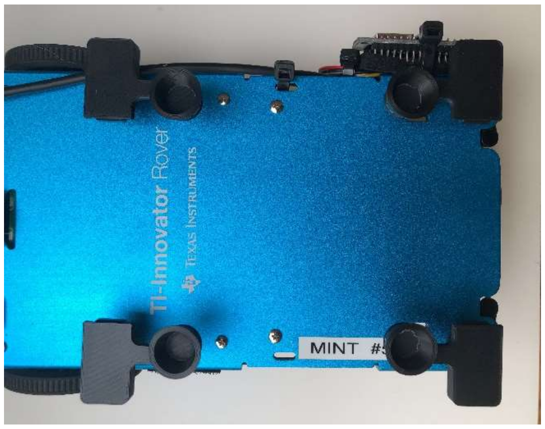


Figure 3 mounting pads on TI-Rover

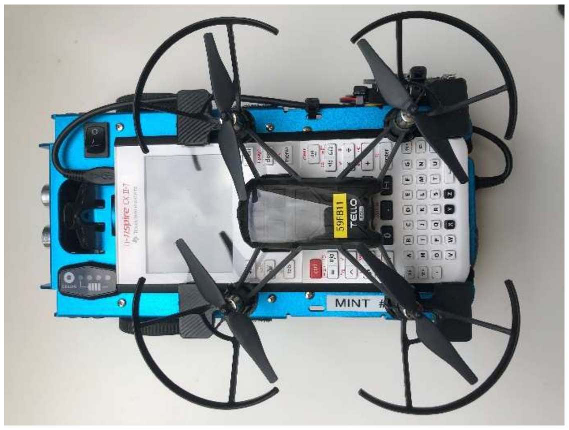


Figure 4 TI-Rover with Nspire CXII and Tello held by mounting pads

Software overview

Please make sure you have your TI-Innovator software updated to the latest version. The current (May 2021) code release is based on the following versions:

- Nspire Desktop Rev 5.3
- TI Hub firmware Rev 1.5
- tello2innovator.ino Rev 1.1
- tello.tns Python module Rev 1.5

The overall software system is divided into three partitions:

1. Arduino UART-to-WiFi interface command line interface (CLI) on NodeMCU
2. Nspire Python tello() module containing all commands necessary to control a Tello or Tello EDU drone
3. Python user code to use tello() module as an object to control a Tello or Tello EDU drone

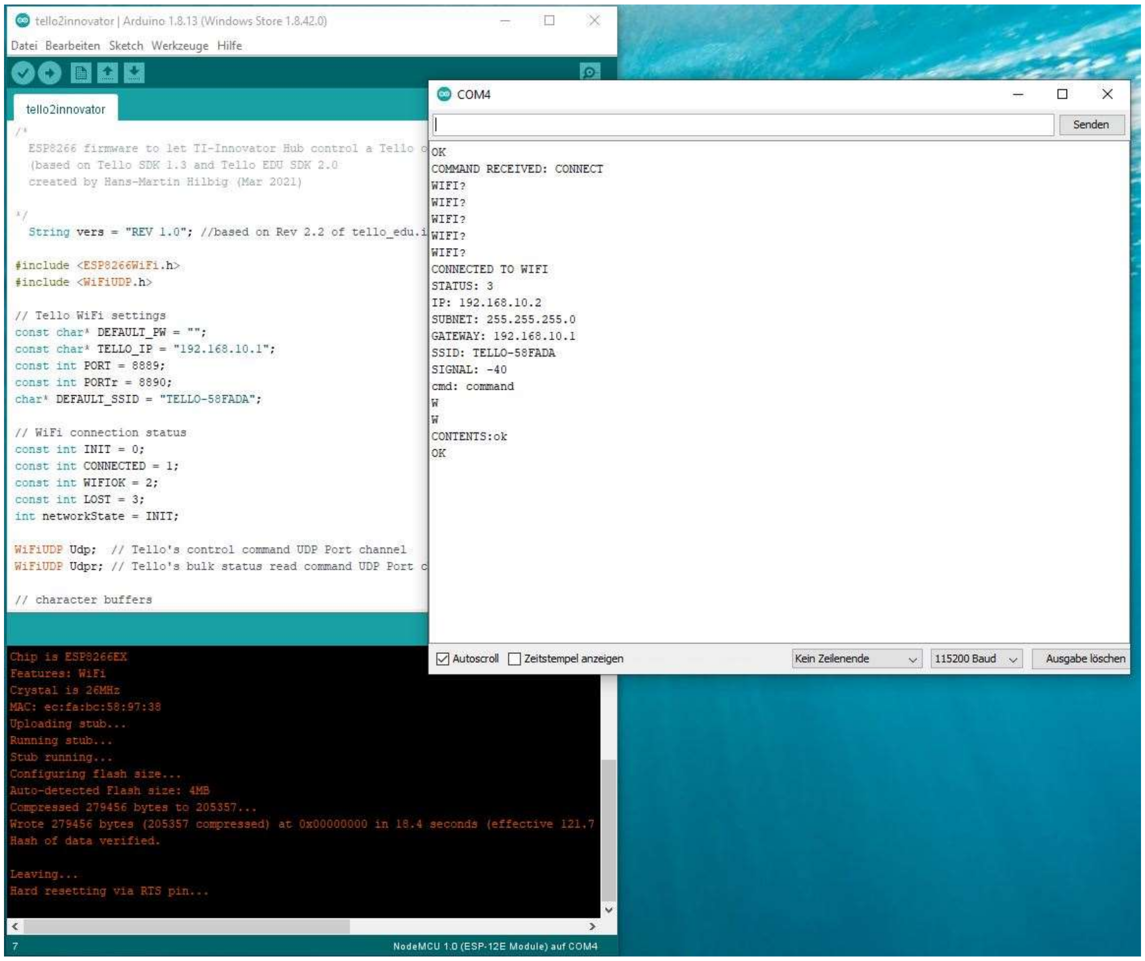
Different to other solutions, the NodeMCU hardware and software provide a standalone interface to allow line-by-line entry or script programming of a Tello drone. The Nspire tello() module creates and transmits Tello SDK scripts to the NodeMCU. Both software partitions have features implemented that help debugging Python user code, detection if Tello already had been connected in a previous code run and an optional failsafe protocol, which would land Tello, stop rotors and stop Python user code if an error is reported by the drone during flight operation.

Arduino tello2innovator.ino software

To transfer the tello2innovator.ino code into NodeMCU, you need to download a copy of the Arduino IDE [5] to your PC. Next, you select NodeMCU v2 from the board manager menu of the Arduino IDE. Finally, connect the USB port of the NodeMCU to your PC. **Make sure, you have your NodeMCU UNPLUGGED from the TI-Hub OUT1 port.** The NodeMCU board will be powered thru the USB interface of the PC. A TI-Hub connected to NodeMCU would act as a second power supply and possibly can damage electronics of the Hub, the NodeMCU or your PC.
Once the tello2innovator.ino code had been successfully loaded into the Flash memory of the NodeMCU, you can do a first connectivity test between NodeMCU and the Tello drone. The Arduino code acts as a standalone command line interface for a Tello drone, without a TI-Hub or Nspire connected.
Here is a step-by-step instruction for testing the WiFi interface between NodeMCU and TI-Hub:

1. Open your Arduino IDE application.
2. Connect the NodeMCU containing the tello2innovator.ino code in its Flash memory to a USB port of your PC. **Do not connect NodeMCU to any other sources (e.g.TI-Hub).**
3. The blue on-board LED of the NodeMCU should blink at a 1Hz frequency, indicating NodeMCU is ready to accept commands.
4. Open the serial monitor of the Arduino IDE to communicate with NodeMCU via a line-by-line interface.
5. Enter the following commands **in UPPERCASE** letters in the command line of the serial monitor, followed by hitting the <enter> key of your PC:
    a. DEBUG <enter>. NodeMCU should respond with <ok>.
    b. SSID abcdef <enter>, where 'abcdef' is the unique 6-digit SSID of your Tello drone. Do not add the 'TELLO-' prefix of the ID, as it is always the same across all Tellos. NodeMCU should respond with <ok>.
    c. Turn on your Tello drone.
    d. CONNECT <enter>. NodeMCU will try to establish a connection with your Tello drone. You should see a couple of 'WIFI?' lines on the serial monitor window, before NodeMCU prints out all WiFi data like shown in figure 5.
    e. NodeMCU should respond with <ok> and you are all set!
    f. Your Tello drone's front LED should blink green, indicating the drone being in SDK command mode. A Tello EDU will blink pink.
    g. If you like, you may check the battery charge of Tello, by typing BAT? <enter>.
    h. Before you actually fly your Tello, please carefully read the Ryzerobotics Safety notes available on the web [8].
    i. You may do a brief takeoff and land of your drone by typing TAKEOFF <enter>. Note the number of 'W' appearing on the serial monitor window. Tello will do a surface pattern recognition check before it responds with a <ok> to NodeMCU. Eventually, the TAKEOFF command will time out, if ambient light is too dim or surface is a unique color without any structure. If TAKEOFF times out, Tello will auto-land.
    j. When you received the <ok> response from the TAKEOFF command, you may type LAND <enter>, to make the Tello land. If no command is sent to Tello after TAKEOFF for more than 15 seconds, Tello will land automatically.
6. In case the above instructions don't execute properly, here's a quick checklist of what might be wrong:
    a. NodeMCU's blue LED is not flashing every other second: Press the <reset> button just left of NodeMCU's USB port. Check if tello2innovator.ino has been loaded into the Flash memory of NodeMCU. Check if there is no other device connected to NodeMCU, except the USB cable to the PC.

b. There is no <ok> received after you entered DEBUG <enter>: maybe you mis-spelled DEBUG? Or you did not type in all UPPERCASE letters? Or you did not press <enter> or the <send> button of the serial monitor? Or you chose the wrong baud rate of the serial monitor (it should be set at 115200 Baud)?
c. After 30 seconds, the CONNECT command times out, responding with an error. Did you enter the correct SSID of your drone before you entered CONNECT? Repeat the SSID command (just enter the 6-digit SSID suffix, not the TELLO- prefix). Is your Tello drone switched on (front line LED flashes yellow on a Tello drone and red on a Tello EDU)? If all this didn't work out, try pressing the <reset> button of NodeMCU, power-off/power-on your drone and start all over again (Debug, SSID, Connect).
d. Tello connection has been established successfully, but drone does not take off. Check for mis-spelling and repeat command. Check for battery capacity using the BAT? command. Battery level should be more than 20% for a take-off.
e. Drone takes off, but does not hover in one place and there is no <ok> response: Ambient light might be too low or texture of the surface is not strong enough for Tello's downward camera pattern recognition system. Try launching Tello from a different location (e.g. a TI-Rover).
f. Drone does not land by command, but rather auto-lands after some time: Tello auto-lands after 15 seconds of no command received. You need to send at least some command (such as BAT?) to let Tello hover in place for more than 15 seconds.


tello2innovator | Arduino 1.8.13 (Windows Store 1.8.42.0)
Datei Bearbeiten Sketch Werkzeuge Hilfe
tello2innovator
COM4
Senden
Autoscroll
Zeitstempel anzeigen
Kein Zeilenende
115200 Baud
Ausgabe löschen
7
NodeMCU 1.0 (ESP-12E Module) auf COM4

```
/*
  ESP8266 firmware to let TI-Innovator Hub control a Tello c
  (based on Tello SDK 1.3 and Tello EDU SDK 2.0
  created by Hans-Martin Hilbig (Mar 2021)

*/
  String vers = "REV 1.0"; //based on Rev 2.2 of tello_edu.i

#include <ESP8266WiFi.h>
#include <WiFiUDP.h>

// Tello WiFi settings
const char* DEFAULT_PW = "";
const char* TELLO_IP = "192.168.10.1";
const int PORT = 8889;
const int PORTr = 8890;
char* DEFAULT_SSID = "TELLO-58FADA";

// WiFi connection status
const int INIT = 0;
const int CONNECTED = 1;
const int WIFIOK = 2;
const int LOST = 3;
int networkState = INIT;

WiFiUDP Udp;  // Tello's control command UDP Port channel
WiFiUDP Udpr; // Tello's bulk status read command UDP Port c

// character buffers
```

```
OK
COMMAND RECEIVED: CONNECT
WIFI?
WIFI?
WIFI?
WIFI?
WIFI?
CONNECTED TO WIFI
STATUS: 3
IP: 192.168.10.2
SUBNET: 255.255.255.0
GATEWAY: 192.168.10.1
SSID: TELLO-58FADA
SIGNAL: -40
cmd: command
W
W
CONTENTS:ok
OK
```

```
Chip is ESP8266EX
Features: WiFi
Crystal is 26MHz
MAC: ec:fa:bc:58:97:38
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 279456 bytes to 205357...
Wrote 279456 bytes (205357 compressed) at 0x00000000 in 18.4 seconds (effective 121.7
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

Figure 5 Arduino IDE with tello2innovator.ino file and serial monitor open

Figure 6 is a block diagram of all software components contained in tello2innovator.ino. All Tello WiFi protocol and timeout management is done by tello2innovator. The command line interface is generic and could be used by any other system capable to communicate over UART at 115kBaud, such as a Raspberry Pi Pico module, a micro:bit board or PC program capable to manage a script over UART.

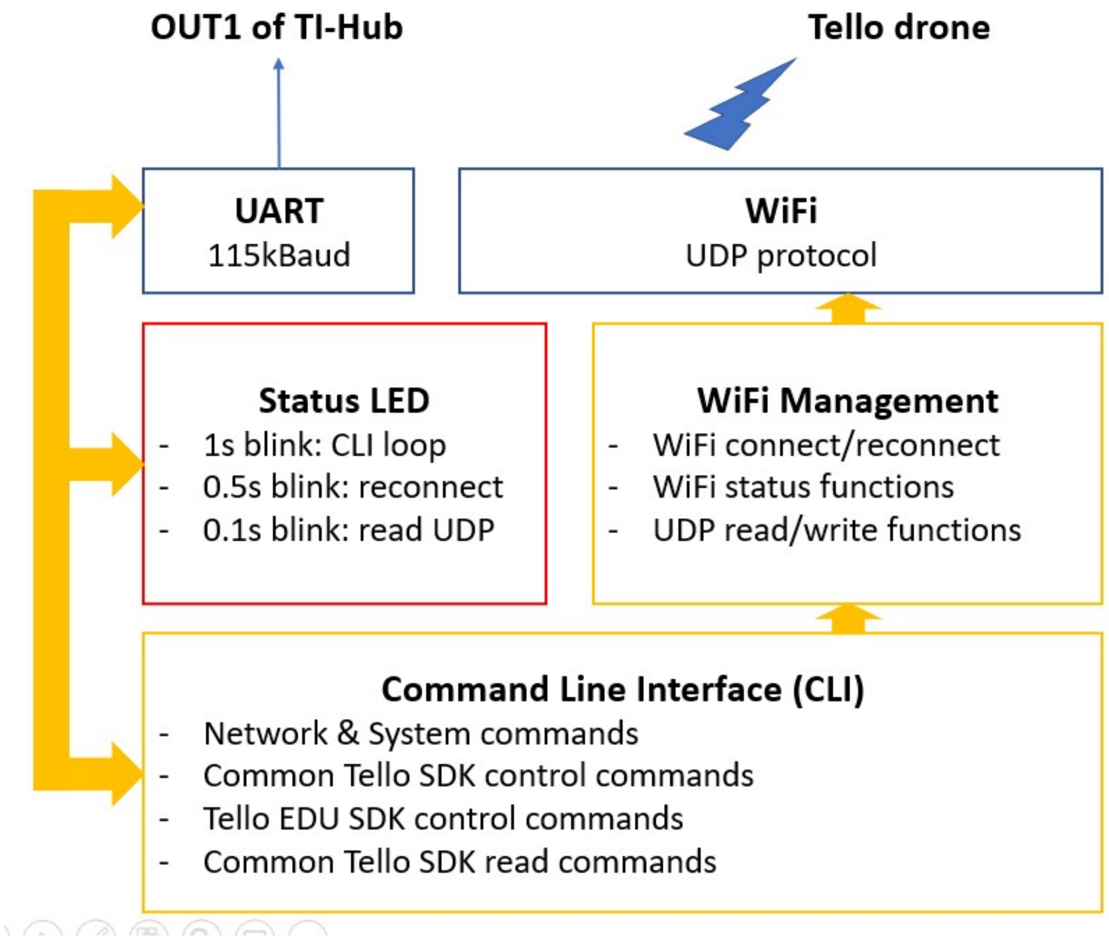


Figure 6 tello2innovator.ino software blocks

tello.tns Nspire Python module

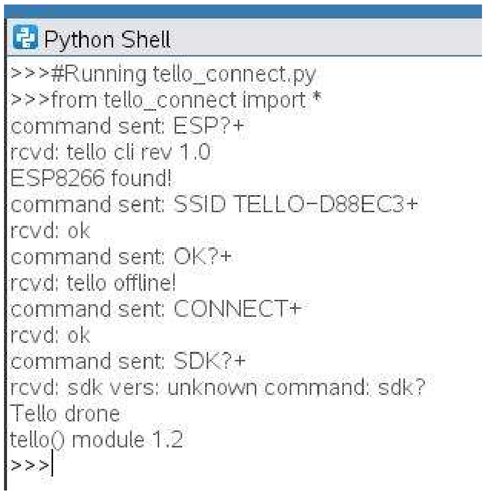
After you checked out your NodeMCU to Tello WiFi interface successfully, it's time to connect the NodeMCU to the TI-Innovator system and do first steps in the Nspire Python environment, using tello() as a Python object.
tello.tns needs to be moved into your PC's PyLib folder and libraries refreshed before tello() is used for the first time.
tello.tns contains three documents:

1.1 A notes page providing a list of all Tello and Tello EDU commands implemented in tello(). Please refer also to Ryzerobotics SDK documentation [6,7] for further details.
1.2 The tello.py code containing the tello() class and all its methods.
1.3 The tello_uart.py module. This module is used by tello() to communicate with the NodeMCU over a UART port available on OUT1 port of the TI-Hub. As there is no official documentation about the TI-Hub UART available from TI so far, some workarounds have been implemented in the UART communication protocol of tello() based on experimental testing. You may hear occasional 'beeps' from TI-Hub during UART communication, but these shouldn't affect proper Tello command transfer to NodeMCU.

tello_test_suite.tns software

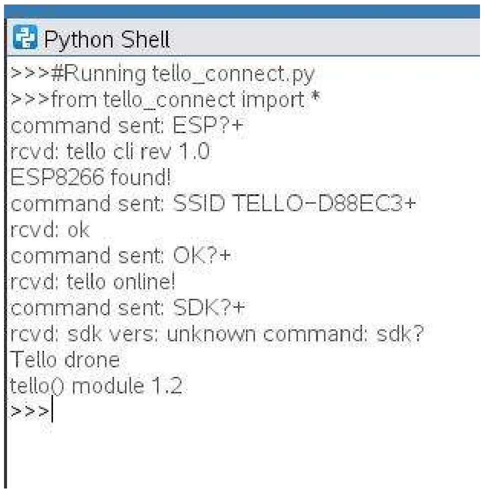
tello_test_suite.tns should help to get familiar with the tello() module (and to check out the entire system). Here is an overview of all Python demo programs provided, along with a description and ideas to try out:

1. tello_connect.py: This is a simple program to check out a successful WiFi connection with your Tello. Insert the 6-digit unique SSID of your drone and run the code. You should see a Shell screen like shown in figure 7. When the tello() object is initialized, the code first checks for the presence of a NodeMCU and if a Tello drone already has been connected before (Tello online). It also checks for a normal Tello or a Tello EDU drone and stores the result in an internal variable. Some of the commands only apply for a Tello EDU drone and will be flagged as an error if executed on a normal Tello drone.



```
>>>#Running tello_connect.py
>>>from tello_connect import *
command sent: ESP?+
rcvd: tello cli rev 1.0
ESP8266 found!
command sent: SSID TELLO-D88EC3+
rcvd: ok
command sent: OK?+
rcvd: tello offline!
command sent: CONNECT+
rcvd: ok
command sent: SDK?+
rcvd: sdk vers: unknown command: sdk?
Tello drone
tello() module 1.2
>>>
```

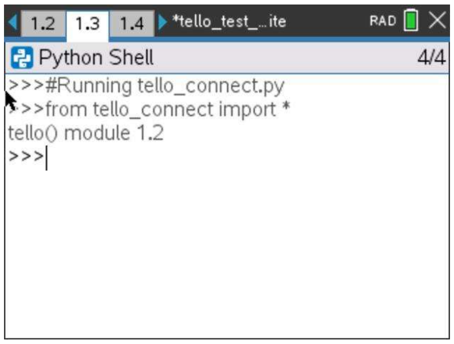
Figure 7 tello_connect, first run



```
>>>#Running tello_connect.py
>>>from tello_connect import *
command sent: ESP?+
rcvd: tello cli rev 1.0
ESP8266 found!
command sent: SSID TELLO-D88EC3+
rcvd: ok
command sent: OK?+
rcvd: tello online!
command sent: SDK?+
rcvd: sdk vers: unknown command: sdk?
Tello drone
tello() module 1.2
>>>
```

Figure 8 tello_connect, second run

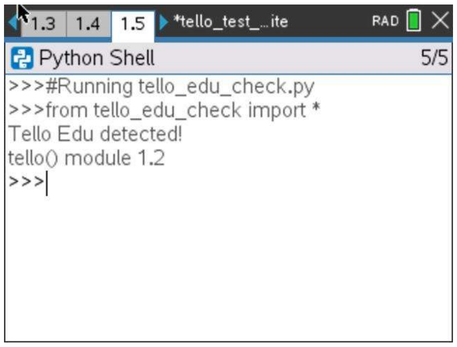
Repeat running the tello_connect code. Now the Shell should look like shown in figure 8. Please note that the drone is reconnected automatically and the connect procedure is skipped. **IMPORTANT:** If you own more than one Tello and like to change from one model to another, you must reset NodeMCU to recognize the new SSID and connect to the new drone. This can be done by either pressing the reset button on NodeMCU or momentarily disconnect the TI-Hub (with NodeMCU connected to OUT1) from your handheld or PC.
Two optional arguments (dbg:Boolean,failsafe:Boolean) can be passed with initializing the tello() object. Both are enabled, i.e. set as True per default. Setting dbg to False would suppress most of the print statements sent from tello() to the Shell. See figure 9 for a screenshot of tello_connect.py executed with tello(False). Setting failsafe to False would disable auto-landing in case of an error received by executing a Tello command. It is recommended to leave failsafe enabled, in order to take benefit from the 2-stage failsafe auto-landing and code stop implemented in tello().



```
>>>#Running tello_connect.py
>>>from tello_connect import *
tello() module 1.2
>>>|
```

Figure 9 tello_connect with dbg disabled (i.e. mydrone = tello(False))

2. tello_edu_check.py: This code example demonstrates the use of the get_sdk() function. get_sdk() returns a True if a Tello EDU drone has been detected, otherwise a False is returned. See a Shell screenshot on figure 10.



```
>>>#Running tello_edu_check.py
>>>from tello_edu_check import *
Tello Edu detected!
tello() module 1.2
>>>|
```

Figure 10 Shell printout of tello_edu_check (Note: dbg has been disabled)

3. takeoff_land.py: This code includes some basic flight control commands. Make sure you start your Tello in an area leaving enough space to hover, move forward and backward and land safely. Rerun the code having changed mydrone = tello(False) and observe Shell print out reduced to a minimum.
4. read_commands.py: Here all Tello read commands are executed. You can execute read commands at any time, after Tello has been connected successfully. Some read commands provide meaningful results only when Tello is in flight. See the Tello SDK manuals [6,7] for details. It’s a good practice to start your Tello Python code with a battery check, to see if enough energy is left to perform the entire flight operation.
5. flysquare.py: Here the Tello flies a 1m square in a counterclockwise direction, starting at its lower right corner. Make sure you have ample space around Tello, as the drone may fly beyond the 1m2 boundaries, caused by inaccuracies of its surface texture based positioning system. Note that Tello flies the square with its nose pointing forward. Try flying the square using .left(), .forward(), .right(), .backward() commands in a clockwise direction, without changing yaw angle at each corner.
6. record_bat_temp.py: This demo shows how to prevent Tello from auto-landing. Code will record battery and temperature status for a given time, e.g. 5 Minutes, while Tello is hovering. Tello will auto-land if no command is received for more than 15 seconds. The time interval for obtaining battery and temperature status has been set to 10s, hence Tello will receive a command within the 15s timeout period. The mydrone.get_bat() command right after takeoff() is needed, as takeoff() typically needs more than 5s to complete. So another command needs to be sent right after takeoff() has completed, to reset the 15s timeout of Tello. See figure 11 for a plot of battery and temperature readings at a 10s interval while hovering for 5 minutes. Be aware that a Tello switched on and just sitting on the ground may get real hot (see

figure 12), as internal electronics are not cooled by the rotating propellers. Switch off Tello when not flying, to save battery and keep Tello's chips cool.

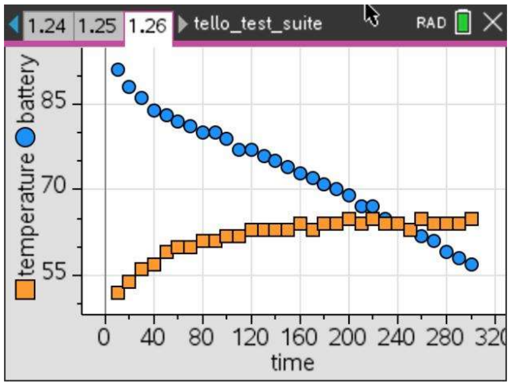


Figure 11 Battery & Temperature graph of Tello hovering for 5 minutes

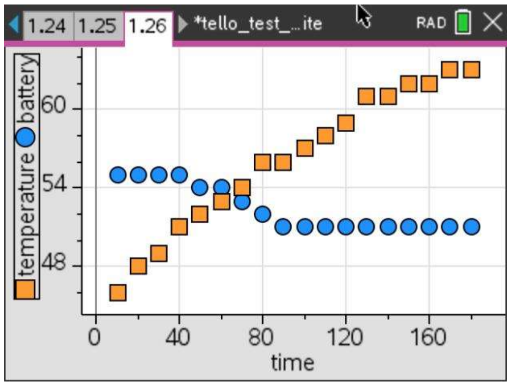


Figure 12 Bat/Temp graph of Tello EDU powered and grounded

7. failsafe_test1.py: This is a demonstration of the failsafe feature built in tello(). Per default, failsafe is enabled when a tello() object is created. Failsafe will scan the response to a command been sent to Tello. If <ok> is received as a response, user code will proceed to the next command. Otherwise, the error response is sent to the Shell, auto-landing will be initiated and Tello will land. User code will proceed, but all subsequent Tello commands will be skipped. Should the auto-landing procedure fail as well, an 'emergency' command is sent to Tello, immediately stopping all motors and user code will be terminated.
   failsafe_test1.py issues a .forward(5) command which will cause an error response from Tello, as the minimum forward distance is 20cm. As failsafe is enabled, tello() will send the error code and location to the Shell and the drone will auto-land. All further tello() related user code will be skipped, but code will not be terminated immediately.
   Similar to switching debug mode <on> or <off> during user code execution, failsafe can also be switched <on> or <off> along code execution. Un-comment the mydrone.set_failsafe(False) line preceding the .forward(5) command. User code will continue with the next tello() commands until the program is finished.
8. go_test.py: Code to demonstrate the 3D-goto command .go(). Please note all arguments are necessary to specify, otherwise an error will be returned.
9. curve_test.py: Code to demonstrate the 3D-arc movement command .curve(). Please see SDK documentation [6,7] for details.
10. curve_circle.py: This code will make Tello fly two half circles in a clockwise direction. Radius of the circle is set to the minimum of 50cm, but allow a lot more airspace, as the .curve() command does not seem overly accurate.
11. mission_pad_test1.py: Last not least there are three code examples to illustrate the use of the mission pad recognition feature of a Tello EDU. Please note that these commands are not executable on a normal Tello and will create an execution error. See Ryzerobotics mission pad user guide for details [9]. To use mission_pad_test1.py place your Tello EDU on the mission pad numbered #1. Orient Tello's nose pointing the same direction as the rocket shown on the mission pad. Code will make Tello take-off, recognize mission pad 1 and fly 1m forward and to the left of mission pad 1, before it returns back to the pad and land.
    Change the orientation of the mission pad by 90 Degrees counterclockwise (rocket pointing to the left) and Tello will fly 1m to the left and back, before it flies another meter back and to the right. Make sure you have enough airspace left for this experiment. This shows how orientation of mission pads changes the orientation of the coordinate system Tello uses as its reference.

12. mission_pad_test2.py: Code shows the use of the .curve() command along with mission pads. Place drone nose in line with rocket nose of the mission pad #1.
13. mission_pad_test3.py: Demoes the use of the jump() command specific to EDU. Place mission pad #1 and mission pad #2 like shown in figure 13 below. Start drone from mission pad #1 and watch it fly over mission pad #2, recognize mission pad #2 and changes its yaw orientation, fly back to mission pad #1, recognize mission pad #1 and change yaw back to the original orientation.
    Purposedly place the wrong mission pad (eg mission pad 6) instead of mission pad #2 and restart program. Tello should fly to mission pad #5, but will fail recognition, as it is expecting mission pad #2 here. Failsafe would trigger, forcing an auto-land and terminate code.

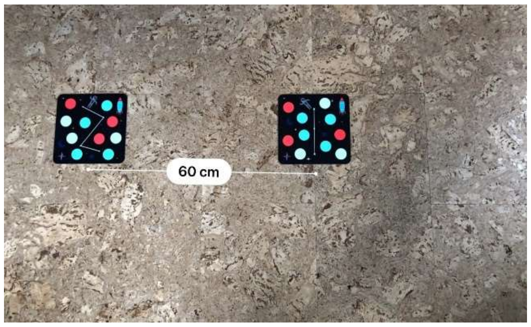


Figure 13 placement of mission pad #1 (right) and pad #2 (left)

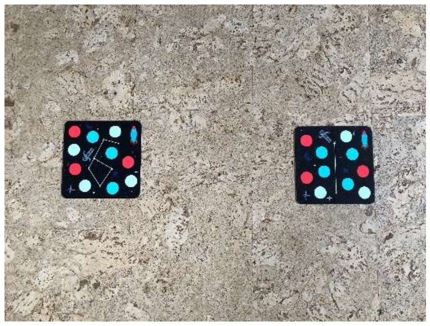


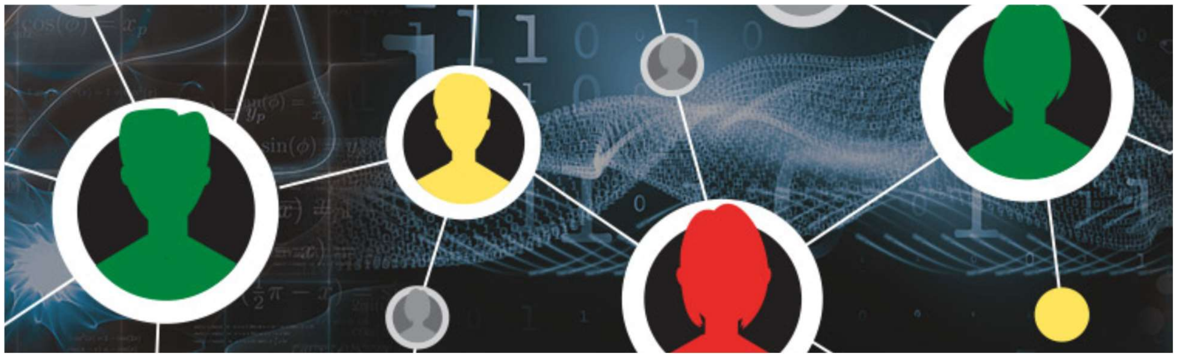
Figure 14 wrong mission pad #6 on the left

Summary:

This document should provide a good overview about the use and possibilities of the tello() module. Using a drone instead of a Rover will make students think in 3-dimensional space, opening up a lot of new opportunities to learn and create.
Although tello2innovator code for NodeMCU and tello() Python code has been tested as thorough as possible, there is no guarantee for code being bug free. But I felt it’s time to disclose it to you as the potential users and I’m happy to receive feedback about your success and failures of using tello() with your students. Shoot me an e-mail to hm-hilbig@web.de and I’ll be happy to help.

Hans-Martin Hilbig, May 2021

Sources:
[1] https://www.seeedstudio.com/NodeMCU-v2-Lua-based-ESP8266-development-kit.html (just connect it to the Arduino GUI and the Lua script will be overwritten)
[2] https://www.seeedstudio.com/Grove-4-pin-Female-Jumper-to-Grove-4-pin-Conversion-Cable-5-PCs-per-PAck.html
[3] https://www.ryzerobotics.com/de/tello
https://www.ryzerobotics.com/de/tello-edu
[4] https://www.thingiverse.com/thing:4858049
[5] https://www.arduino.cc/en/software
[6] https://terra-1-g.djicdn.com/2d4dce68897a46b19fc717f3576b7c6a/Tello%20%E7%BC%96%E7%A8%8B%E7%9B%B8%E5%85%B3/For%20Tello/Tello%20SDK%20Documentation%20EN_1.3_1122.pdf
[7] https://dl-cdn.ryzerobotics.com/downloads/Tello/Tello%20SDK%202.0%20User%20Guide.pdf
[8] https://dl-cdn.ryzerobotics.com/downloads/Tello/20180211/Tello+Disclaimer+and+Safety+Guidelines+(EN)+v1.0.pdf
[9] https://dl-cdn.ryzerobotics.com/downloads/Tello/Tello%20Mission%20Pad%20User%20Guide.pdf



T³ DEUTSCHLAND

Teachers Teaching with Technology™