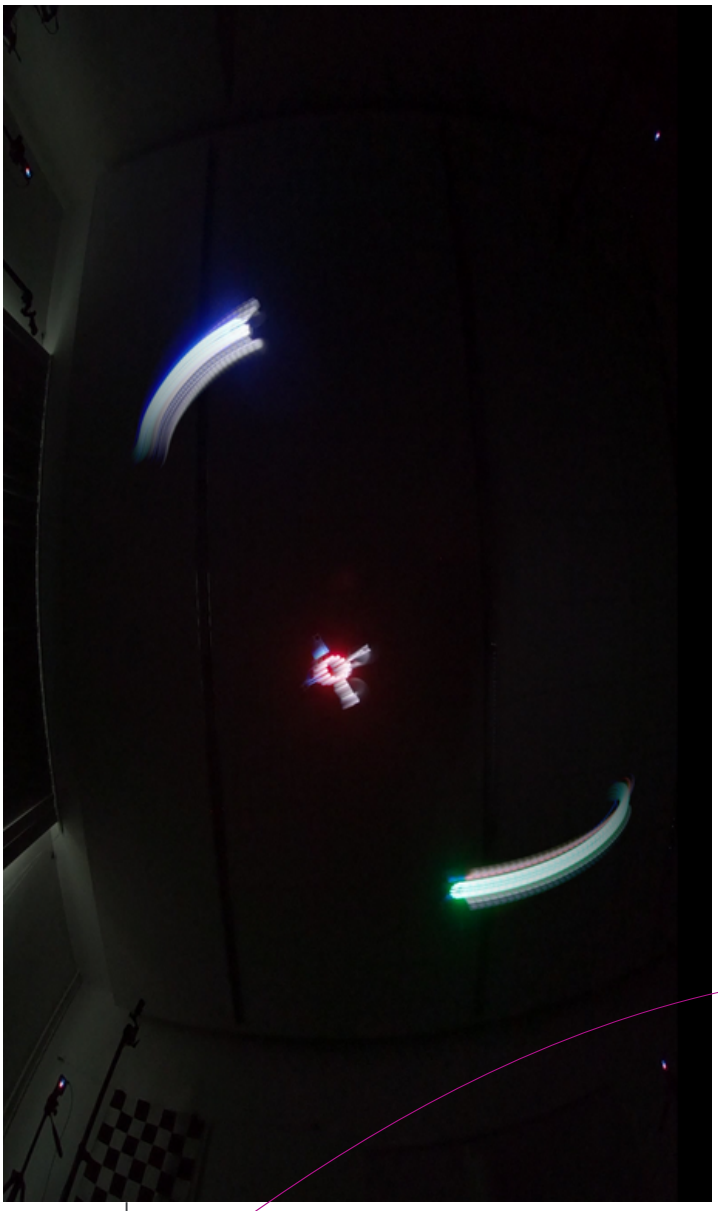
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN NORMANDIE



# Crazyflie Drone Swarms

## Automation via CSV

Loïck Toupin

January 2022 - May 2023

Module - ITI - Year 4

# Contents

# 1 Introduction

This document is a user manual for flying **Crazyflie** drones from the **Bitcraze** brand in a swarm. I have been working with these drones for the past three semesters as part of a project during my studies in Computer Science at INSA. I have exclusively used the drones with the Loco positioning system, with eight beacons.

This document provides instructions for the program I have developed. It allows for improved swarm flights. The main improvement is the use of .csv files, which automate and store flights with greater precision. This also helps in anticipating possible inter-drone collisions and visualizing flights in advance through simulation.

## 1.1 Installation

First, you need to install the cfclient software and the drivers for the radio controller (follow the instructions: Installation). Then, you can power on and connect a drone, and verify that the horizon in the cfclient matches the drone's tilt. If you encounter orientation issues, consider updating cfclient and the drones.
To connect a drone, follow these steps:

- Plug in the USB antenna.
- Locate the number written beneath the drone.
- In the cfclient, enter the address E7E7E7E70n (where n is the drone number) and radio://80/2M.
- Press the scan button.
- Then click on connect.

If the accelerometer is working fine, everything is good! To power off and on the drones, there is a small button located at the front to avoid damaging the battery cables.

Figure 1: Front LEDs in operation

## 2 Quick Start

To get quick results, here's how to launch example flights. In the following sections, we will explain how to create your own custom choreographies.

- First, navigate to the **4_gif** folder. Choose a simple flight to start with, for example, *circles_2_drones*.
- Look for the folder with the same name in **3_OUTPUTCSV**, where you will find the starting positions of the drones.

  In our case, take two drones whose numbers are indicated underneath. Sort them in ascending order according to their numbers and place them accordingly.

  Here's an explanation: the drone numbers in the CSV file all start at 0. For a two-drone flight, there are positions for drones 0 and 1 in the file. If we choose drones 4 and 6, we need to place drone 4 at the position of drone 0 in the file, and drone 6 at the position of drone 1 in the file.
- Open **csvCommander.py**, go to line 31, and change the flight name in the *file* variable. In our case, set it to "circles_2_drones".
- Uncomment the URI addresses of the drones being used (for example, 4 and 6) and comment out the others. The last character in the address corresponds to the number beneath each drone.
- Plug in the USB antenna, ensure that the drones and the beacons are powered on, navigate your terminal to the **1_code** folder, and run **csvComander.py**.

# 3 Flight Programming

## 3.1 CSV Structure

A line in the CSV represents a step in the flight at a certain time $t$ and contains the following information:

| **Column** | **Detail** | **Domain** |
|---|---|---|
| Step | Step of the flight | Natural numbers |
| Id | Drone number | Natural numbers |
| Command | Type of command | {'Takeoff', 'Land', 'Goto', 'Ring', 'Headlight'} |
| X-R-L | Depending on **Command**:<br>• if Goto: drone's X-coordinate<br>• if Ring: red value of the LED<br>• if Headlight: state of the front LED | • if Goto: Real number (meters)<br>• if Ring: Integer (8 bits)<br>• if Headlight: Integer (0 or 1) |
| Y-G | • if Goto: drone's Y-coordinate<br>• if Ring: green value of the LED | • if Goto: Real number (meters)<br>• if Ring: Integer (8 bits) |
| Z-B | • if Goto: drone's Z-coordinate<br>• if Ring: blue value of the LED | • if Goto: Real number (meters)<br>• if Ring: Integer (8 bits) |
| Yaw-Intensity | • if Goto: drone's yaw rotation<br>• if Ring: LED intensity | • if Goto: Real number (radians)<br>• if Ring: Real number (0.0...1.0) |
| Duration | Step duration | Real number in seconds (often 0.042: 1/24) |

Figure 2: CSV Structure Details

If a real number is entered when an integer is expected, it will be rounded to the nearest integer.

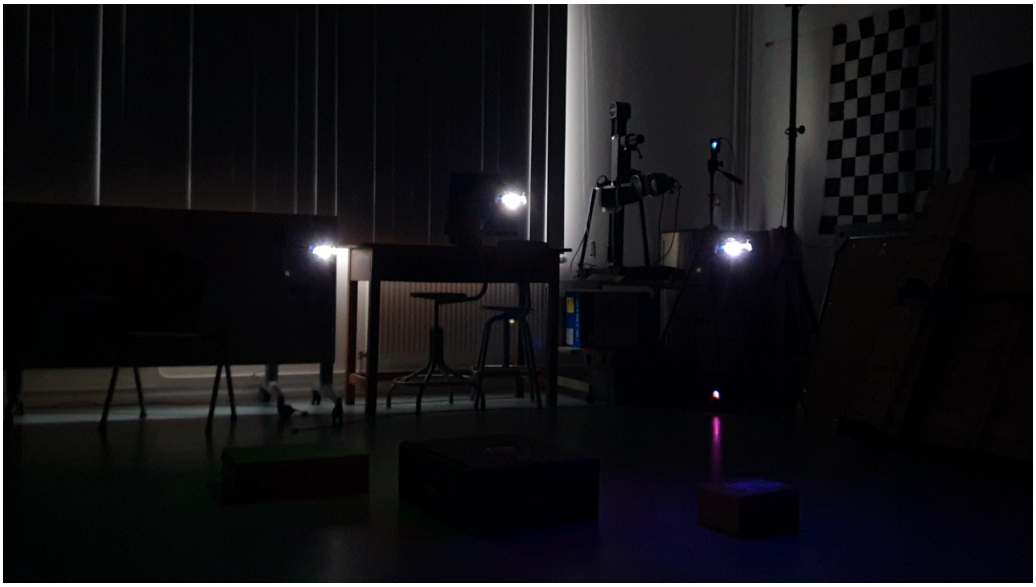


Figure 3: Operating front LEDs

### 3.2 Constructing a CSV File

We will create multiple CSV files, for example, one per drone per command. The takeoff and landing commands will be added automatically.

In each function, we provide the following information:

- **id**: the drone number to differentiate between drones. Assigning command addresses to drones will be done later.
- **first_step**: the step number at which the maneuver begins. For example, we can create a circle between steps 1 and 50, and later with the same drone ID, create another circle between steps 100 and 150. Therefore, we create two separate CSV files for these two maneuvers using the same function.
- **duration** in seconds.
- **offset** in x, y, z, and yaw, which defines the starting position of the maneuver. CSV merging will be performed later.

It is important to keep in mind that we work at a frequency of 24 steps per second. In the majority of cases, we enter 0.042 seconds (1/24) in the *duration* column mentioned above.

```
1  Step,Id,Command,X-R-L,Y-G,Z-B,Yaw-Intensity,Duration
2  0.0,0.0,Takeof,0.0,1.0,1.0,0.0,2.0
3  1.0,0.0,Goto,0.0,1.0,1.0,0.0,0.042
4  2.0,0.0,Goto,-0.026,1.0,1.0,0.0,0.042
5  3.0,0.0,Goto,-0.052,0.999,1.0,0.0,0.042
6  4.0,0.0,Goto,-0.078,0.997,1.0,0.0,0.042
7  5.0,0.0,Goto,-0.105,0.995,1.0,0.0,0.042
8  6.0,0.0,Goto,-0.131,0.991,1.0,0.0,0.042
9  7.0,0.0,Goto,-0.156,0.988,1.0,0.0,0.042
10 8.0,0.0,Goto,-0.182,0.983,1.0,0.0,0.042
11 9.0,0.0,Goto,-0.208,0.978,1.0,0.0,0.042
12 10.0,0.0,Land,0.0,1.0,0.0,0.0,2.0
13
```

Figure 4: Sample CSV for a drone

### 3.2.1 Goto Model

Below is an adaptable model for the drone movement commands. An example, the formation of circles, is explained on the next page.

The parts to be completed (enclosed by comments) are:

- 1 the function signature
- 2 the constants
- 3 the variables that depend on the step and the calculation of the coordinates at each step

```python
def goto_model(id=0, first_step=1, duration=10, offset=[0, 0, 1, 0], fps=24, folder="example", extension='part1'):
    # 1 add specific parameters to the function signature

    steps = fps * duration
    delta_t = round(duration / steps, 3)
    header = ['Step', 'Id', 'Command', 'X-R-L', 'Y-G', 'Z-B', 'Yaw-Intensity', 'Duration']

    # ----------------2 my specific constants-------------------

    # ----------------------------------------------------------------

    if not os.path.exists('../2_csv/' + folder):
        os.makedirs('../2_csv/' + folder)
    file = file_name(folder=folder, mode='circle', drone=id, extension=extension)
    with open('../2_csv/' + folder + '/' + file, 'w') as file:
        for i in range(len(header) - 1):
            file.write(str(header[i]) + ',')
        file.write(str(header[i + 1]) + '\n')  # prevents the last comma
        for i in range(steps + 1):
            # --------3 variables dependent on step------------------
            # complete without considering offsets
            x = 1
            y = 1
            z = 1
            yaw = 1
            changed = 1  # set to zero if no change:
            # the step (identical) will not be created
            # ----------------------------------------------------------------
            # do not modify
            if changed:
                changed = 0
                step = [first_step + i, id, "Goto", round(offset[0] + x, 3), round(offset[1] + y, 3),
                        round(offset[2] + z, 3), round(offset[3] + yaw, 3), delta_t]
                for i in range(len(step) - 1):
                    file.write(str(step[i]) + ',')
                file.write(str(step[i + 1]) + '\n')  # prevents the last comma
```

Listing 1: goto_model functiun

### 3.2.2 Goto example - Circle

Here is an example of a complete Goto function: the circle.

The function signature, constants, and variables have been filled in.

```
def circle(id=0, first_step=1, duration=10, offset=[0, 0, 1, 0], fps=24, folder="example", radius=1, angle_init=0,
           direction=1, axes=['x', 'y'], extension=""):
    steps = fps * duration
    delta_t = round(1 / fps, 3)
    header = ['Step', 'Id', 'Command', 'X-R-L', 'Y-G', 'Z-B', 'Yaw-Intensity', 'Duration']

    # ---------------my specific constants-----------------------
    # complete
    delta_angle = math.pi * 2 / steps
    # -----------------------------------------------------------
    # do not modify
    if not os.path.exists('../2_csv/' + folder):
        os.makedirs('../2_csv/' + folder)
    file = file_name(folder=folder, mode='circle', drone=id, extension=extension)
    with open('../2_csv/' + folder + '/' + file, 'w') as file:
        for i in range(len(header) - 1):
            file.write(str(header[i]) + ',')
        file.write(str(header[i + 1]) + '\n')  # prevents the last comma
        for i in range(steps + 1):
            # ------------------------------variables dependent on step------------------------
            # complete without considering offsets
            sin = round(radius * math.sin(angle_init + (-direction) * delta_angle * i), 3)
            cos = round(radius * math.cos(angle_init + (-direction) * delta_angle * i), 3)
            if sorted(axes) == ['x', 'y']:
                x = sin
                y = cos
                z = 0
                yaw = 0
            elif sorted(axes) == ['y', 'z']:
                x = 0
                y = cos
                z = sin
                yaw = 0
            elif sorted(axes) == ['x', 'z']:
                x = cos
                y = 0
                z = sin
                yaw = 0
            else:
                print("error axes")
                break
```

```
42              changed = 1  # set to zero if no change
43              # ------------------------------------------------------------
44              # do not modify
45              if changed:
46                  changed = 0
47                  step = [first_step + i, id, "Goto", round(offset[0] + x, 3), round(
    offset[1] + y, 3),
48                          round(offset[2] + z, 3), round(offset[3] + yaw, 3), delta_t]
49                  for i in range(len(step) - 1):
50                      file.write(str(step[i]) + ',')
51                  file.write(str(step[i + 1]) + '\n')  # prevents the last comma
52          print("last step: {}".format(first_step + steps))
53          return first_step + steps
```

Listing 2: Circle example

To go further, we can make functions that call this one by varying the parameters:

```
1 def circles_several_drones(nb_drones=3, first_id=0, duration=10, first_step=1,
2 \axes=["y", "x"], angle_init=0, radius=0.5,direction=1,
3 \folder="example", offset=[0, 0, 0.5, 0], extension=""):
4
5     for i in range(nb_drones):
6         last_step = circle(id=first_id + i, first_step=first_step, duration=duration,
      axes=axes,angle_init=angle_init + i * (math.pi * 2 / nb_drones), direction=
    direction, radius=radius,offset=offset, folder=folder, extension=extension)
7     return last_step
```

Listing 3: Circle_drones example

Here, we vary the initial position using *angle_init* and the drone number.

Running **circle_several_drones** creates three CSV files, one per drone.

### 3.2.3 Example RGB LEDs

The operation is similar here, but we specify the color and intensity values. Note the use of **changed=0** to only create steps that are different from the previous ones.

```
def rgb_model(id=0, first_step=0, duration=10, fps=24, folder="example", extension="part1"):
    # 1. Add my specific function parameters to the function signature

    steps = fps * duration
    delta_t = round(duration / steps, 3)
    header = ['Step', 'Id', 'Command', 'X-R-L', 'Y-G', 'Z-B', 'Yaw-Intensity', 'Duration']

    # ----------------2. My specific constants-------------------
    r = 0
    g = 0
    b = 0
    intensity = 1
    duration = 0  # fade duration, 0 for hard
    # -----------------------------------------------------------

    # Do not modify
    if not os.path.exists('../csv/' + folder):
        os.makedirs('../csv/' + folder)
    file = file_name(folder=folder, mode='rgb', drone=id, extension=extension)
    with open('../csv/' + folder + '/' + file, 'w') as file:
        for i in range(len(header) - 1):
            file.write(str(header[i]) + ',')
        file.write(str(header[i + 1]) + '\n')  # prevent the last comma
        for i in range(steps + 1):
            # --------3. Variables dependent on the step------------------
            # Complete without considering offsets
            r = 1
            g = 1
            b = 1
            changed = 1  # set to zero if no change: the step (identical) will not be created
            # -----------------------------------------------------------
            # Do not modify
            if changed:
                changed = 0
                step = [first_step + i, id, "Ring", round(r), round(g), round(b), round(intensity, 3), duration]
                for i in range(len(step) - 1):
                    file.write(str(step[i]) + ',')
                file.write(str(step[i + 1]) + '\n')  # prevent the last comma
        print("Last step: {}".format(first_step + steps))
        return first_step + steps
```

Listing 4: Example RGB LEDs

### 3.2.4 Headlight example

```
def headlight(id=0, first_step=1, duration=10, fps=24, folder="example", extension="part1", modulo=24):
    steps = fps * duration
    delta_t = round(1 / fps, 3)
    header = ['Step', 'Id', 'Command', 'X-R-L', 'Y-G', 'Z-B', 'Yaw-Intensity', 'Duration']

    # -----------------My specific constants-------------------------------------
    # Complete as needed
    led = 0
    # ---------------------------------------------------------------------

    # Do not modify
    if not os.path.exists('../2_csv/' + folder):
        os.makedirs('../2_csv/' + folder)
    file = file_name(folder=folder, mode='headlight', drone=id, extension=extension)
    with open('../2_csv/' + folder + '/' + file, 'w') as file:
        for i in range(len(header) - 1):
            file.write(str(header[i]) + ',')
        file.write(str(header[i + 1]) + '\n')
        for i in range(steps + 1):
            # ---------------Variables dependent on the step-----------------
            # Complete as needed
            if i % modulo == 0:
                led = 1 - led  # 1 for on, 0 for off
                changed = 1  # set to zero if no change
            duration = 0  # duration of the lighting (unused if led=0), set to 0 to turn on without time limit
            # -----------------------------------------------------------

            # Do not modify
            if changed:
                changed = 0
                step = [first_step + i, id, "Headlight", led, 0, 0, 0, duration]
                for i in range(len(step) - 1):
                    file.write(str(step[i]) + ',')
                file.write(str(step[i + 1]) + '\n')  # prevent the last comma
        print("Last step: {}".format(first_step + steps))
        return first_step + steps
```

Listing 5: Exemple headlight

## 3.3 Merging CSV Files

Once all the flight CSV files are created, they need to be merged into a single file. The function **prepare_csv()** adds takeoff commands before the first "goto" command of each drone and landing

commands after the last "goto" command of each drone. When there are gaps between two sets of "goto" commands for the same drone, the trajectory is linearly interpolated.

Finally, a 3D plot GIF file can be created using the **csv2gif** function, where the front LED is represented by a yellow ring around the point.

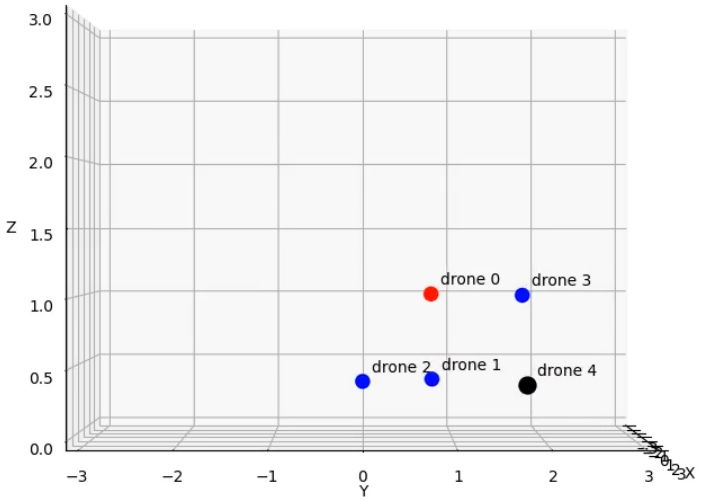



Figure 5: Flight Simulator

And that's it! Now that you've seen how to create flight trajectories, follow this protocol.

## 3.4 Flight Creation Protocol

### 3.4.1 CSV Creation

- Write your movement and LED functions.
- Call them in the **csvBuilder.py** script.
- Choose the flight name on line 25.
- Navigate to the **1_code** directory in your terminal.
- Run **csvBuilder.py**.

The final CSV file is created in the **3_OUTPUTCSV** folder in a subfolder with the chosen name.

### 3.4.2 Flight Execution

Refer to section 2 for flight execution instructions.

# 4 Conclusion and Acknowledgments

The December video still illustrates the current progress: **here (YouTube)**.

This report is the culmination of three semesters of work on this project for me. It has pushed me to my limits and confirmed my desire to continue in the field of autonomous vehicles.

Firstly, I would like to thank Benoit Gaüzere, who made this project possible and provided guidance. I also want to thank Florine Chevrier, Mathias Hersent, and Raphaël Largeau, my group from the first semester, through which we quickly understood the operation of Crazyflie drones and created our first choreographies.

Finally, I would like to thank Roman Mouhawej, Paul-Adrien Marie, Arthur Deberne, and Hugo Boulenger for their participation this semester, and I wish them a lot of fun with the project if they continue with it next year.