Working with Crazyflies on Ubuntu

Running Crazyflie program:

```
$ python3 -m cfclient.gui
```

Crazyflie tutorial:

Communication **radio://0/80/2M/E7E7E7E7E7.**

Crazyradio PA
- Crazyradio Real-Time Protocol (CRTP)

Unique URI
- Medium: Which radio you are using
- Channel: 10-120
- Communication Speed: bits per second
- Address: of the individual Crazyflie drone

Broadcast to multiple Crazyflies
- Sure, as long as you are on the same channel.

Internal Measurement Unit (IMU):
Accelerometers: Acceleration in x,y,z
Gyroscope: Angular movement
Pressure Sensor: Used for height estimation
(barometer is off by default)

Crazyflie can carry 15 grams maximum.
Flow deck can fly up to 4 meters or 13 feet.
Flow deck has a 30x30 pixel camera to detect pixel flow
4.2 degree field of view
Crazyflie’s initial coordinates are wherever the Crazyflie starts.

Swarm control:
Broadcasting
- Messages with no return expected

Multiple Crazyflies per Crazyradio
Use same channel
Can not send specific trajectories at each time step
Crazyflies need to do more themselves

Flashing multiple Crazyflies
- Bit off-topic but this will need to be done in advance.
- Go to examples/demos/swarm_demo
- Don't forget to put in geometry!
    - Maybe this shifted during setting up
- Use the .sh script for flashing all of them
    - ./cload_all.sh

Crazyflie Movement:

```
With SyncCrazyflie(URI) as scf:
     With MotionCommander(scf) as mc:
```

When using the flow deck for localization:

The drone will maintain height relative to what is underneath the flow deck. The flow deck has trouble maintaining stability when moving fast especially in circles.

https://www.bitcraze.io/documentation/repository/crazyflie-lib-python/master/api/cflib/positioning/motion_commander/

MotionCommander(crazyflie, default_height=0.3)
Immediately take off at default_height (meters) and enable movement commands.
Parameters:
- crazyflie: a Crazyflie or SyncCrazyflie instance
- default_height: The default height to fly at

Class variables
- RATE
- VELOCITY

Basic movement:

def **move_disntance**(self, distance_x_m, distance_y_m, distance_z_m, velocity=0.2)
Move in a straight line. (+) X is forward (+) Y is left (+) Z is up.
- distance_x_m: The distance to travel along the X-axis (meters)
- distance_y_m: The distance to travel along the Y-axis (meters)
- distance_z_m: The distance to travel along the Z-axis (meters)
- velocity: The velocity of the motion (meters/second)

def **start_linear_motion**(self, velocity_x_m, velocity_y_m, velocity_z_m, rate_yaw=0.0)
Start a linear motion with an optional yaw rate input. This function returns immediately.
- velocity_x_m: The velocity along the X-axis (meters/second)
- velocity_y_m: The velocity along the Y-axis (meters/second)
- velocity_z_m: The velocity along the Z-axis (meters/second)
- rate_yaw: The angular rate (degrees/second)

Start/stop:

def **take_off**(self, height=None, velocity=0.2)
Tekes off, that is starts the motors, goes straight up and hovers. Do not call this function if you use the keyword. Take off is done automatically when the context is created.
- height: The height (meters) to hover at. None uses the default height set when constructed.
- velocity: The velocity (meters/second) when taking off

def **stop**(self)
Stop any motion and hover.

def **land**(self, velocity=0.2)
Go straight down and turn off the motors.
Do not call this function if you use the with keyword. Landing is done automatically when the context goes out of scope.
- velocity: The velocity (meters/second) when going down

Move up/down:

def **up**(self, distance_m, velocity=0.2)
Go up
   - distance_m: The distance to travel (meters)
   - velocity: The velocity of the motion (meters/second)

def **start_up**(self, velocity=0.2)
Start moving up. This function returns immediately.
   - velocity: The velocity of the motion (meters/second)

def **down**(self, distance_m, velocity=0.2)
Go down
   - distance_m: The distance to travel (meters)
   - velocity: The velocity of the motion (meters/second)

def **start_down**(self, velocity=0.2)
Start moving down. This function returns immediately.
   - velocity: The velocity of the motion (meters/second)

Move forward/back:

def **forward**(self, distance_m, velocity=0.2)
Go forward
   - distance_m: The distance to travel (meters)
   - velocity: The velocity of the motion (meters/second)

def **start_forward**(self, velocity=0.2)
Start moving forward. This function return immediately.
   - velocity: The velocity of the motion (meters/second)

def **back**(self, distance_m, velocity=0.2)
Go backwards
   - distance_m: The distance to travel (meters)
   - velocity: The velocity of the motion (meters/second)

def **start_back**(self, velocity=0.2)
Start moving backwards. This function returns immediately.
   - velocity: The velocity of the motion (meters/second)

Left movements:

def **left**(self, distance_m, velocity=0.2)
Go left
- distance_m: The distance to travel (meters)
- velocity: The velocity of the motion (meters/second)

def **start_left**(self, velocity=0.2)
Start moving left. This function return immediately.
- velocity: The velocity of the motion (meters/second)

def **turn_left**(self, angle_degrees, rate=72.0)
Turn to the left, staying on the spot.
- angle_degrees: How far to turn (degrees)
- rate: The turning speed (degrees/second)

def **start_turn_left**(self, rate=72.0)
Sart turning left. This function returns immediately.
- rate: The angular rate (degrees/second)

def **circle_left**(self, radius_m, velocity=0.2, angle_degrees=360.0)
Go in circle, counter clockwise
- radius_m: The radius of the circle (meters)
- velocity: The velocity along the circle (meters/second)
- angle_degrees: How far to go in the circle (degrees)

def **start_circle_left**(self, radius_m, velocity=0.2)
Start a circular motion to the left. This function returns immediately
- radius_m: The radius of the circle (meters)
- velocity: The velocity of the motion (meters/second)

Right movements:

def **right**(self, distance_m, velocity=0.2)
Go right
- distance_m: The distance to travel (meters)
- velocity: The velocity of the motion (meters/second)

def **start_right**(self, velocity=0.2)
Start moving right. This function returns immediately.
- velocity: The velocity of the motion (meters/second)

def **turn_right**(self, angle_degrees, rate=72.0)
Turn to the right, Staying on the spot
- angle_degrees: How far to turn (degrees)
- rate: The turning speed (degrees/second)

def **start_turn_right**(self, rate=72.0)
Start turning right. This function returns immediately.
- rate: The angular rate (degrees/second)

def **circle_right**(self, radius_m, velocity=0.2, angle_degrees=360.0)
Go in circle, clockwise
- radius_m: The radius of the circle (meters)
- velocity: The velocity along the circle (meters/second)
- angle_degrees: How far to go in the circle (degrees)

def **start_circle_right**(self, radius_m, velocity=0.2)
Start a circular motion to the right. This function returns immediately
- radius_m: The radius of the circle (meters)
- velocity: The velocity of the motion (meters/second)