

# Getting Started with Crazyflie Development

---

Required hardware you should have: Crazyflie 2.0 quadcopter, Xbox (or generic) controller and a Crazyradio.

## 1 INSTALLATION

There are several things that must be installed on your PC before you can start development on the crazyflie quadcopter.

- 1 Virtual Box: <https://www.virtualbox.org/wiki/Downloads>
- 2 Virtual Box Extension Pack: (same as above)
- 3 Crazyflie VM: <https://www.bitcraze.io/getting-started-with-the-crazyflie-2-0/>

Detailed instructions on how to install the software is documented on the websites. Once you have the Crazyflie VM installed on virtual box, log in to the Crazyflie VM and run the following commands from terminal (ctrl-alt-T).

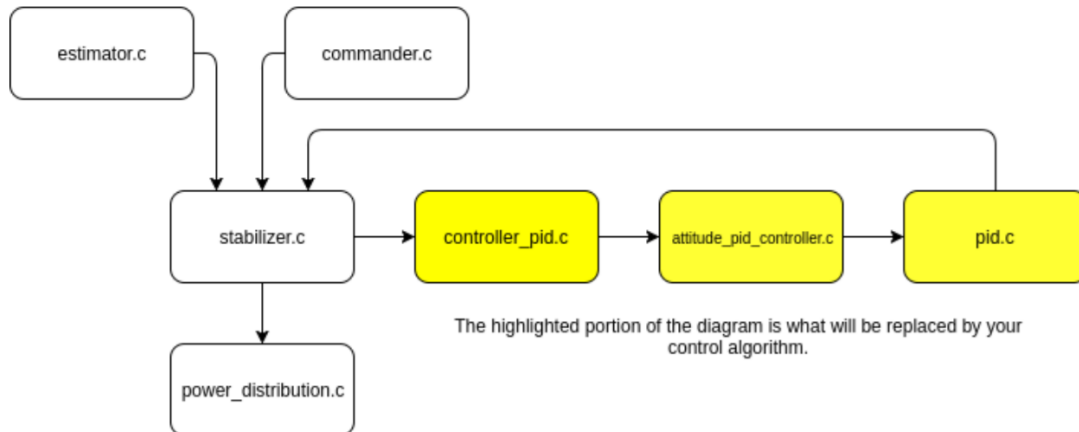
```
git clone https://github.com/Peter-Jan/crazyflie_ws
cd crazyflie_ws
chmod +x install.sh
./install.sh
```

NOTE: The default user is 'bitcraze' and password is 'crazyflie'.

## 2 CRAZYFLIE VM

From within the Crazyflie VM is where we will be doing the majority of the embedded controls algorithm. For the scope of this course we will focus only on the controls portion of the code. All of which will be implemented in C on the Crazyflie. Once your Crazyflie VM is set up, click on "Update all projects" icon from the desktop. After you have updated the firmware 1 on your VM to the latest version you may launch Eclipse from within the VM and start modifying the code.

The structure of the default control loop is as shown below:



Commander is where the Crazyflie will get the control setpoint from the teleop. Estimator reads the sensors and Kalman filters the data to give Euler angles which are then fed through to the stabilizer. The stabilizer will call on the PID controller (which you will replace). Finally the PID controller will return the control signal to stabilizer, which will call on power distribution to send a PWM signal to the motors. The PWM signal is set with a linear mapping that takes a unsigned 16 bit integer value from 0% to 100%.

The main control loop is contained within `stabilizer.c`. This loop is ran at a rate of 1 KHz. When you are satisfied with your code and wish to upload it onto the Crazyflie, from within the Eclipse IDE, go to "Make Target" tab on the right hand side. Under crazyflie-firmware click:

- 1 clean
- 2 make CLOAD
- 3 flash using radio

NOTE: Your crazyflie must be in bootloader mode in order to flash the firmware onto it. Power off the device, then press and hold the power button for 3 seconds to enter bootloader mode.

\*If you are comfortable using linux you can bypass the VM and develop on linux natively! (<https://github.com/bitcraze/crazyflie-firmware>)

### 3 ROBOT OPERATING SYSTEM(ROS)

If you are unfamiliar with ROS it is recommended that you go through the tutorial to get a basic understanding of what ROS does. (<http://wiki.ros.org/ROS/Tutorials>).

The file that you cloned from Github is the workspace that has the necessary packages to fly the drone by teleoperation. It also contains the package required to use the optical tracking system.

To fly your drone simply start ROS by running the command 'roscore' in terminal. Open a new terminal and run 'roslaunch crazyflie\_tools scan' this will print the uri address for the crazyflie onto console (ex. radio://0/20/2M). You will copy this address into the launch file for teleoperation. This launch file should be located at (/path\_to\_your\_workspace/src/crazyflie\_ros/crazyflie\_demo/launch/teleop\_xbox360.launch).

```
<launch>
<arg name="uri" default="radio://0/20/2M" />
<arg name="joy_dev" default="/dev/input/js0" />

<include file="$(find crazyflie_driver)/launch/crazyflie_server.launch">
</include>

<group ns="crazyflie">
  <!--<param name="crazyflie/pid_rate/yaw_kp" value="200"/>-->
  <include file="$(find crazyflie_driver)/launch/crazyflie_add.launch">
    <arg name="uri" value="$(arg uri)" />
    <arg name="tf_prefix" value="crazyflie" />
    <arg name="enable_logging" value="True" />
  </include>

  <node name="joy" pkg="joy" type="joy_node" output="screen" >
    <param name="dev" value="$(arg joy_dev)" />
  </node>

  <include file="$(find crazyflie_demo)/launch/xbox360.launch">
  </include>

  <node name="crazyflie_demo_controller" pkg="crazyflie_demo" type="controller.py" output="screen">
  </node>
</group>

<node pkg="rviz" type="rviz" name="rviz" args="-d $(find crazyflie_demo)/launch/crazyflie.rviz" />
<!--<node pkg="rqt_plot" type="rqt_plot" name="rqt_plot" args="/crazyflie/temperature/temperature"/> -->
<node pkg="rqt_plot" type="rqt_plot" name="rqt_plot2" args="/crazyflie/battery"/>
<node pkg="rqt_plot" type="rqt_plot" name="rqt_plot3" args="/crazyflie/rssi"/>
</launch>
```

Now run 'roslaunch crazyflie\_demo teleop\_xbox360.launch' in terminal and you should be flying your drone!

\*If joystick appears unresponsive make sure that you are linking the correct one. Under where you modified the uri for the radio you have to select the proper joystick ex: '/dev/input/jsX'.

\*\*You can run 'jstest /dev/input/jsX' and test the input from your joystick.

## 4 DATA LOGGING

To log data that is on the Crazyflie it must be added to the logging table. Looking at the bottom of `stabilizer.c` (crazyflie-firmware) you will find that many different signals are already being logged. The argument being passed to `LOG_GROUP_START()` is the topic and the second argument for `LOG_ADD` is the variable.

```
163
164
165 LOG_GROUP_START(tester)
166 LOG_ADD(LOG_UINT32, myTick, &myTick)
167 LOG_GROUP_STOP(tester)
168
169 LOG_GROUP_START(ctrltarget)
170 LOG_ADD(LOG_FLOAT, roll, &setpoint.attitude.roll)
171 LOG_ADD(LOG_FLOAT, pitch, &setpoint.attitude.pitch)
172 LOG_ADD(LOG_FLOAT, yaw, &setpoint.attitudeRate.yaw)
173 LOG_GROUP_STOP(ctrltarget)
174
```

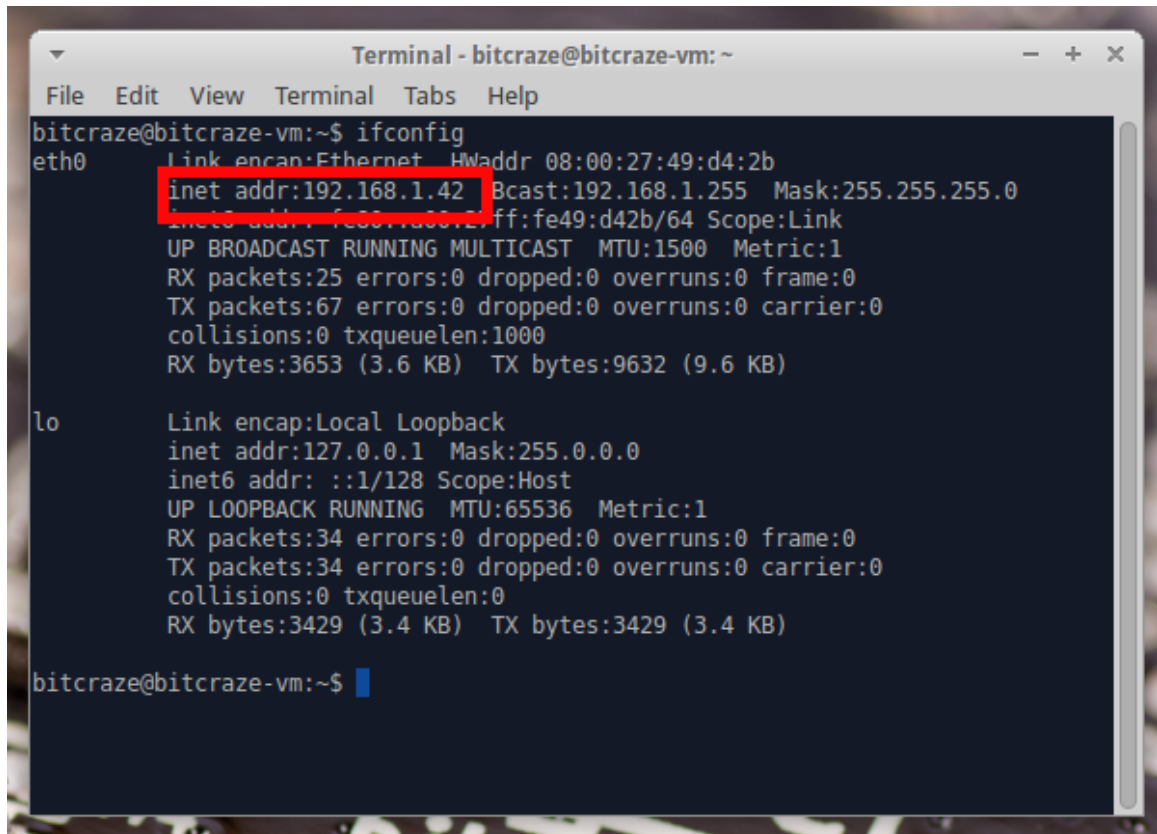
Once you have added the variable to be logged you must flash the crazyflie with the modified firmware. To begin data logging to file modify `crazyflie_add.launch`.

```
16
17 <!-- can make arguments to sub into param below or simply replace $(xxx) with topic -->
18 <arg name="log_topic" default="stabilizer"/>
19 <arg name="log_topic1" default="tester"/>
20
21
22 <node pkg="crazyflie_driver" type="crazyflie_add" name="crazyflie_add" output="screen">
23   <param name="uri" value="$(arg uri)"/>
24   <param name="tf_prefix" value="$(arg tf_prefix)"/>
25   <param name="roll_trim" value="$(arg roll_trim)"/>
26   <param name="pitch_trim" value="$(arg pitch_trim)"/>
27   <param name="enable_logging" value="$(arg enable_logging)"/>
28   <param name="use_ros_time" value="$(arg use_ros_time)"/>
29   <param name="enable_logging_imu" value="$(arg enable_logging_imu)"/>
30   <param name="enable_logging_temperature" value="$(arg enable_logging_temperature)"/>
31   <param name="enable_logging_magnetic_field" value="$(arg enable_logging_magnetic_field)"/>
32   <param name="enable_logging_pressure" value="$(arg enable_logging_pressure)"/>
33   <param name="enable_logging_battery" value="$(arg enable_logging_battery)"/>
34
35   <!-- The topics below must be separated by commas -->
36   <rosparam param="genericLogTopics" value="[stabilizer, tester]" />
37   <rosparam param="genericLogTopicFrequencies" value="[10, 10]" />
38   <rosparam param="genericLogTopic" value="$(arg log_topic)" Variables="[stabilizer.roll, stabilizer.pitch, stabilizer.yaw]" />
39   <rosparam param="genericLogTopic" value="$(arg log_topic1)" Variables="[tester.myTick]" />
40
41 </node>
42 </launch>
43
44
45
```

The modified `crazyflie_ros` package will save the logged topic into comma delimited text files located at `(~/ros)`

## 5 OPTITRACK

In order to use Optitrack with the VM make sure you go to settings in Virtual Box and select Network -> Bridged Adapter. Make sure that your PC is connected to the 'HDR-network' (The password is written on the router). Once that is complete start the VM and run 'ifconfig' and take note of the VM's IP address.

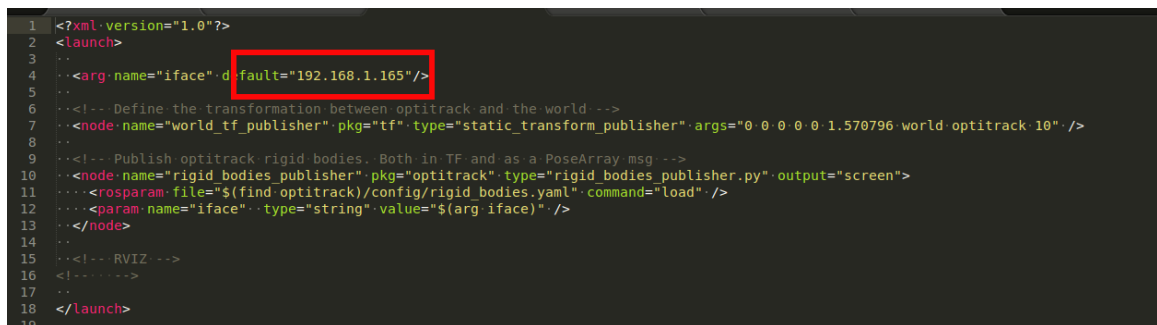


```
Terminal - bitcraze@bitcraze-vm: ~
File Edit View Terminal Tabs Help
bitcraze@bitcraze-vm:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:49:d4:2b
          inet addr:192.168.1.42  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:12:ff:fe49:d42b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:25 errors:0 dropped:0 overruns:0 frame:0
          TX packets:67 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3653 (3.6 KB)  TX bytes:9632 (9.6 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:34 errors:0 dropped:0 overruns:0 frame:0
          TX packets:34 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3429 (3.4 KB)  TX bytes:3429 (3.4 KB)

bitcraze@bitcraze-vm:~$
```

Next, modify the `optitrack_pipeline.launch` file to include the IP of your VM. This file should be located at `(/crazyflie_ws/src/optitrack/launch/)`.



```
1 <?xml version="1.0"?>
2 <launch>
3 ..
4 ..<arg name="iface" default="192.168.1.165"/>
5 ..
6 ..<!-- Define the transformation between optitrack and the world -->
7 ..<node name="world_tf_publisher" pkg="tf" type="static_transform_publisher" args="0 0 0 0 1.570796 world optitrack 10" />
8 ..
9 ..<!-- Publish optitrack rigid bodies. Both in TF and as a PoseArray.msg -->
10 ..<node name="rigid_bodies_publisher" pkg="optitrack" type="rigid_bodies_publisher.py" output="screen">
11 ..<roscpp param file="$(find optitrack)/config/rigid_bodies.yaml" command="load" />
12 ..<param name="iface" type="string" value="$(arg iface)" />
13 ..</node>
14 ..
15 ..<!-- RVIZ -->
16 ..<!-- .. -->
17 ..
18 </launch>
19
```

To operate the optical tracking system, start Motive software on the PC in the optical tracking lab. From the quick start choose Open Existing Project. Select 'Crazyfly\_CBF'.

Now, in the Motive software go to View -> Data Streaming. Under Advanced Network Options -> Multicast Interface, enter the IP address of your virtual machine.

When you have completed the above, start the ROS node with command

```
roslaunch optitrack optitrack_pipeline.launch
```

The node will save the position and the attitude of the crazyfly to (~/.ros) as comma delimited text files.

## 6 OTHER USEFUL LINKS

- 1 Thrust transfer function: <https://wiki.bitcraze.io/misc:investigations:thrust>
- 2 Crazyflie Python API: <https://github.com/bitcraze/crazyflie-lib-python>
- 3 Bitcraze Wiki: <https://wiki.bitcraze.io/index>
- 4 Optitrack: <https://github.com/crigroup/optitrack>
- 5 Crazyflie ROS: [https://github.com/whoenig/crazyflie\\_ros](https://github.com/whoenig/crazyflie_ros)
- 6 Crazyflie firmware: <https://github.com/bitcraze/crazyflie-firmware>