Crazyflie 2.0 Drone Stabilization

Project Number: 18-2- 1- 1590

By: Laurene Bouskila, Shir Barlas Advisor: Yoni Mandel

Project Carried Out at TAU Ventures

Iby and Aladar Fleischman Faculty of Engineering Tel Aviv University

הפקולטה להנדסה ע"ש איבי ואלדר פליישמן אוניברסיטת תל-אביב

XTEND
Extending Reality Skywards

Introduction

The field of Unmanned Flying Vehicles (UAVs) research and applications has grown rapidly over the past few years. Amongst it is the usage and research of quadropters, such as the Crazyflie. Most of the work in this area uses controllers that are derived from linearization of the model around hover conditions and are stable only under reasonably small angles.The drone can support several sensors such as the optical flow (measures height), IMU sensors, accelerometer, barometer amongst others… These sensors give us access to important datas which enable us evaluating the stability of the device through the Client hardware.

Goals

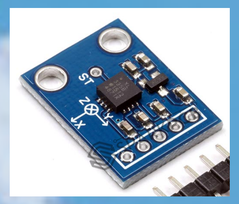
Improve the stabilisation of the quadcopter when it's given an impulse start, random angles, height …

Get to the python code implementing the client and modify it in order to achieve steady state of the device.

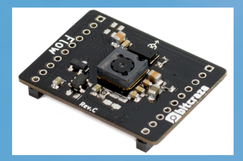
Methods

To reach these goals, we had to go though several steps. As can be seen on the scheme depicted hereafter, the datas are received from the different sensors before being send to the Kalman estimator. After that, they are converted to inertial frame thanks to a set of equations the Kalman filter obeys to. They are then sent to the Mellinger controller and to the rest of the system.

To reach the former goals, we had to understand the working process of the Kalman estimation filter and of the Mellinger controller as well as their implementation in the code and the parts of the client code that were needed to be improved.



accelerometer

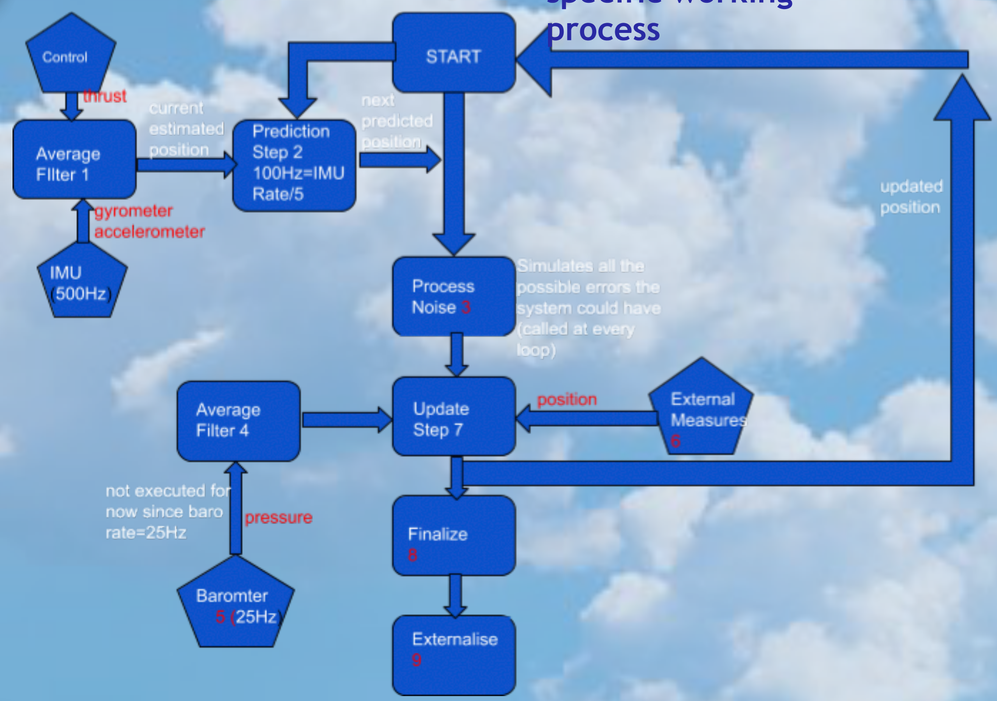


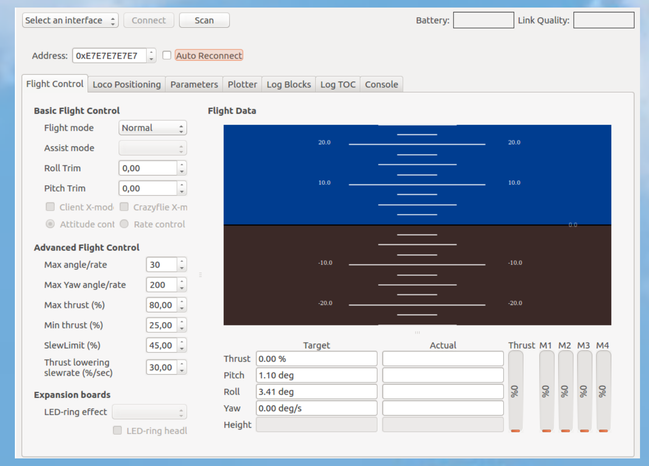
optical flow

Datas received through the sensors are read and analysed through the client hardware

Different types of sensors

The Kalman Estimator obeys to a specific working process





The Client is implemented by a python code project

```
        pk = CRTPPacket()
        pk.port = CRTPPort.COMMANDER_GENERIC
        pk.data = struct.pack('<Bffff', TYPE_ZDISTANCE,
                              roll, pitch, yawrate, zdistance)
        self._cf.send_packet(pk)

    def send_hover_setpoint(self, vx, vy, yawrate, zdistance):
        """
        Control mode where the height is send as an absolute setpoint (intended
        to be the distance to the surface under the Crazyflie).

        vx and vy are in m/s
        yawrate is in degrees/s
        """
        pk = CRTPPacket()
        pk.port = CRTPPort.COMMANDER_GENERIC
        pk.data = struct.pack('<Bffff', TYPE_HOVER,
                              vx, vy, yawrate, zdistance)
        self._cf.send_packet(pk)
```

The client code communicates with the Crazyflie and we test the stabilisation flight sensitivity by analysing the quadcopter flight.