Technical review of quadcopter firmware design by Dan Coleman.

Introduction:

A quadcopter is a type of UAV (Unmanned Aerial Vehicle) that has four arms, there is a motor attached to a propeller in each arm. Two of the rotors turn clockwise, while the other two turn counter clockwise. Quadcopters require a flight computer to produce the desired motions by converting commands to Rotations Per Minute (RPM) of the propeller as they are aerodynamically unstable. Since the life uses rotors (vertically standing propellers) they are considered rotorcrafts. [1]

Part 1: Overview of Quadrotor hardware and control:

The Controller:

Typically a quadcopter comes with a 4-channel controller that sends commands affect its throttle (increasing or decreasing the altitude of the quadcopter), yaw (pointing the front of the copter in a different direction), pitch (tilting the copter back or forward), roll (tilting the copter left or right), [2] 2.4 GHz is the most common communication frequency used by controllers. This is also the typical frequency used for WIFI connections. The controller also has 4 “trim” buttons that allow minor changes to the quadcopter flight pattern. If you notice that with no control input, the drone drifts in a particular direction, applying trim for that control input in the opposite direction will remove the drift.[1]

Actuators:

An actuator is a hardware device that is responsible for converting a controller command signal into a change in physical parameters. This change is typically a mechanical change (for example position or velocity). The most important actuators are the Motor and the Propellers.

Motors are what are used to make the **propellers** spin. It is important to remember that there are two sets of propellers, and two set of motors. Two of the motors spin clockwise while the other set spin counter-clockwise. With these two sets rotating in opposite directions the total angular momentum is zero. [3] The pitch on the two sets of propellers are different. So, if you switch a set of opposite propellers, they will be blowing air up instead of down. This results in a downward force on the quadcopter causing the drone to flip instead of flying.[1]

Sensors:

Sensors are one of the most important parts of a quadcopter, while thrust can be generated simply by the motor and propeller stable flight require several sensors. A sensor being a something that measures continuous changes in an environment and converts this to digital data. In this section we shall look at a few.

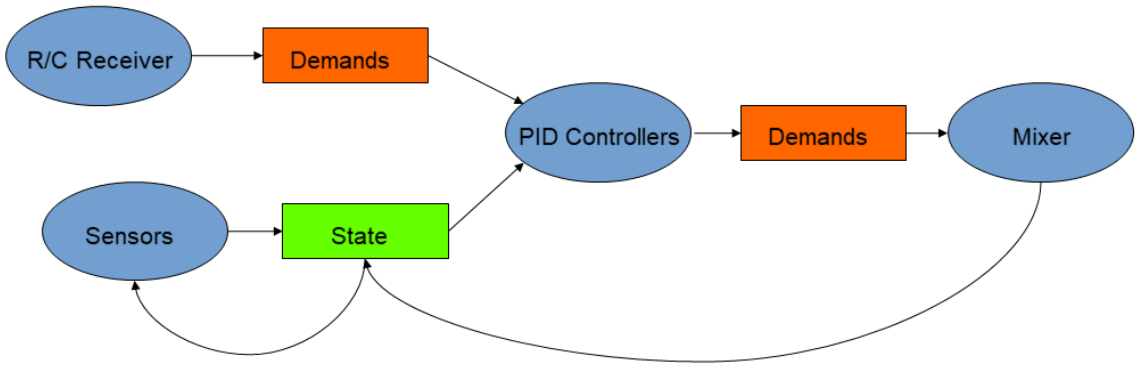
Accelerometer’s measure the lateral acceleration of the quadcopter on the X, Y and Z axis. This is used to determine the position and orientation of the quadcopter in flight and play a key role in flight control. Due to the sensitivity of the sensors they play a key role in stabilising the quadcopter, as they are more immune to vibrations than other technologies and minimize problems from the vibrations generated by the movement of the rotating motors and propellers. [4] **Gyroscope**’s measure the rotational acceleration of the sensor allowing them to determine the rate of change to the pitch, roll and yaw of the quadcopter. The change in angle information is used to provide stability to the quadcopter and prevent it from wobbling. [5] The gyroscope and accelerometer together are called the Microelectromechanical system (MEMS). MEMS is used to detect sudden stops such as vehicle crashes along with helping keep the drone flying in a stable manner. The

Magnetometer which measures the magnetic field around the quadcopter helping determine its orientation. Some quadcopters also have Pressure Sensors called **Barometers** to measure air pressure to help determine its altitude, as air pressure decreases as the machine flies higher, this is not the most common sensor however.[6] Finally we have the **Quaternion.** A quaternion is a four-element vector that can be used to encode any rotation in a 3D coordinate system. [7]

Part 2: Models and firmware design:

Firmware is the code that runs on the quadcopter's controller. There are many different options of firmware, while there isn't a standardised typing, firmware is typically either considered either Flight Controllers or Autopilots. **Autopilots** are typically considered firmware with a system enabling the quadcopter to fly autonomously to way-points. These include Global Positioning Systems that allow a quadcopter to measure distance by measuring how long a signal takes to travel from a satellite. These potentially allow quadcopters to fly on their own from take-off to landing. While **Flight Controllers** do not include these instead just assisting in quadcopter stability.[5]

The firmware we will look at today is a **Flight Control** toolkit **Hackflight**. Hackflight is a C++ toolkit for building multirotor flight controllers. There are two basic data types in Hackflight: **state** and **demands**. State is the state of the vehicle at a given time (altitude, orientation, angular velocity) which are modified by the sensors (gyroscope, accelerometer, barometer, etc.), then once this state is determined the Controller modifies the demands given to the quadcopter (throttle, roll, pitch and yaw). The demands are then sent to a **mixer** which determines each value to be sent to each motor, which spins the propellers and modifies the state. [8]



Sensors:

The minimum sensor readings needed for Hackflight are gyroscope and quaternion, however it also supports accelerometers, magnetometers and barometers. All these readings are subclasses of the SurfaceMountSensor class which itself is a subclass of the Sensor class. The Sensor class is an abstract class specifying two methods that any sensor must Implement:

1. Reporting if the sensor is ready to deliver new data
2. Modifying the quadcopters state.

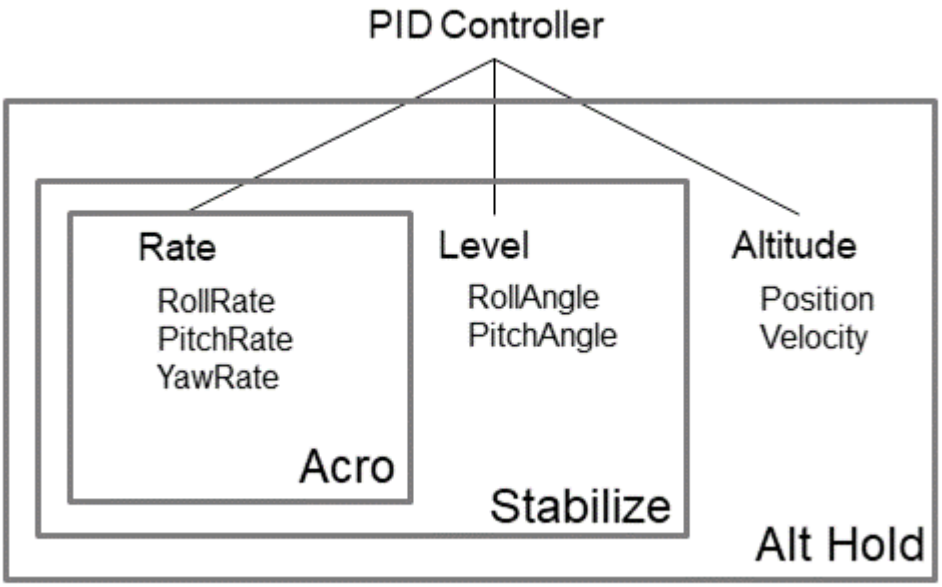
By having each sensor report its readiness it avoids having to write separate timing loops for each sensor in the main loop of the code. To implement additional sensors you directly sub-class the Sensor class and call addSensor() to add it to the checkOptionalSensors method.

Controllers:

Similar to sensors, Hackflight's Proportional-integral-derivative (PID) controllers are subclasses of an abstract class. The current state and demands are taken in using the modifyDemands() method, and the demands are modified based on the state. As with sensors, you can sub-class the PidController class to ensure that your PID controller is called.

The two most important points about PID controllers in Hackflight are that:

1. A PID controller is not the same as a flight mode.
2. It matters in which order you add PID controllers, because the output of one PID controller is the input to the next. [8]



References

1. *A Beginner's Guide to Multirotor Systems & Flight Proficiency*. (n.d.). Retrieved from https://uavcoach.com/how-to-fly-a-quadcopter-guide
2. *What is a Quadcopter*. (n.d.). Retrieved from https://www.droneomega.com/what-is-a-quadcopter/
3. *How a quadcopter works with propellers and motors*. (n.d.). Retrieved from https://www.dronezon.com/learn-about-drones-quadcopters/how-a-quadcopter-works-with-propellers-and-motors-direction-design-explained/
4. *How Many Sensors are in a Drone, And What do they Do?* (n.d.). Retrieved from https://www.fierceelectronics.com/components/how-many-sensors-are-a-drone-and-what-do-they-do
5. *Beginners guide to drone autopilots (flight controllers) and how they work*. (n.d.). Retrieved from https://www.dronetrest.com/t/beginners-guide-to-drone-autopilots-flight-controllers-and-how-they-work/1380

6. *Quadcopters: Sensors*. (n.d.). Retrieved from https://hoverbear.org/blog/quadcopters-sensors/

7. *Understanding Quaternions*. (n.d.). Retrieved from http://www.chrobotics.com/library/understanding-quaternions

8. *Hackflight's github*. (n.d.). Retrieved from https://github.com/simondlevy/Hackflight