ISSN: 1813-4890

Pixhawk Flight Control System Architecture Analysis

Zishuai Wang[a], Guoliang Zou[b], and Jiehong Huang[c]

Taishan University, Taian, 271000, China

[a]1514370697@qq.com, [b]1835716463@qq.com, [c]1556714436@qq.com

Abstract

Pixhawk is a stand-alone project designed to provide high-end industry-standard autopilot hardware for academia, hobby, and industry at low cost and high availability. It is widely praised by drone enthusiasts for its powerful and abundant hardware. The Pixhawk supports both APM and PX4 firmware, and here we do a software architecture analysis of the native PX4 firmware. I will carry out the system architecture analysis of Pixhawk from four aspects, namely: hardware architecture introduction, PX4 development environment construction, PX4 software architecture analysis and summary.

Keywords

Pixhawk; Flight Control; Architectur.

1. Introduction

Pixhawk is a 32-bit open source flight control based on the ARM chip, originally using a split design of px4 (composed of px4fmu and px4io components), which was later merged into a whole to form the present pixhawk. Its hardware and software are open source, so many different hardware and software versions are derived, and the original distributor is 3D Robotics in the United States. The PIXHAWK is a high-performance autopilot suitable for fixed wing, multirotor, helicopters, cars, ships as well as any of the other mobile robotic platforms. It targets high-end research, amateur and industry needs, and combines the features of PX4FMU + PX4IO. The shape of the Pixhawk is shown in Figure 1.

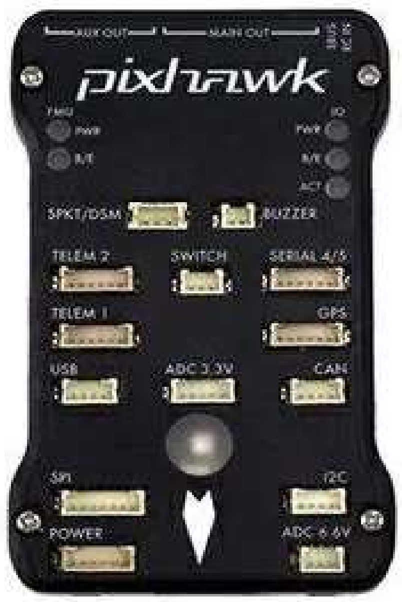

Figure 1. The shape of the Pixhawk

2. Hardware Architecture Analysis

The 32-bit STM32F427 microprocessor utilized by the Pixhawk has an FPU, a Cortex M4 kernel, a 168MHz main frequency, 252MIPS of processing power, 256KB of RAM, 2MB of flash memory, and an STM32F103 fault coprocessor chip. The MicroL3GD20H 16-bit gyroscope and MicroLSM303D 14-bit accelerometer/magnetometer form ST Company, the MPU 6000 3-axis accelerometer/gyroscope from Invensense Company, and the MS5611 barometer from MEAS Company serve as the sensors. In addition, it includes a very powerful hardware interface, including Spektrum DSM/DSM2/DSM-X satellite compatible input interface, FutabaS, with 5 UART (serial ports), including one with high power, 2 with HW flow control, with 2 CAN (one with internal 3.3V transceiver, one with extended connector). BUS is compatible with a variety of interfaces, including input and output, PPM and signal input, RSSI (PWM or voltage) input, I2C, SPI, ADC, internal and external microUSB interface, and more. Any industrial robot platform can benefit from it courtesy to its rich interfaces. Pixhawk's power system and protection features include an ideal diode with automatic failover, the ability to adapt to high power servo motors (up to 10V) and large currents (10A+), overload protection for all peripheral ends, and ESD protection with all inputs. The Pixhawk has three power sources, allowing for triple power source redundancy. The three power sources are the USB inputs, servo motor contribution, and power supply box input.It uses the battery voltage out from battery pack package first, which has a voltage range of 4.8 to 5.4 volts, so the energy input out from servo motor, which has a voltage range of 4.8 to 5.4 volts as well. The voltage range is exactly the same if you to choose USB generator input.

The STM32F427 chip serves as the primary microcontroller in the Pixhawk hardware platform and is used to receive and send information from any and all sensors, settle posture, and analyse and manipulate other data. The power system manages the supply of power for the pic microcontroller and all module in addition to providing power with electromechanical adjustment.Comparing this fault coprocessor is intriguing. The cocontroller of the aircraft can sense when the central controller of the airframe is flying or when other errors result in the motor being not managed. Then it takes over control of the motor and works it in a safe mode until the main controller is detected to be controlling the motor once again.

3. PX4 Development Environment Construction

The PX4 code can be developed on Mac OS, Linux or Windows, and it is recommended on Mac OS and Linux, because image processing and advanced navigation are not easy to develop on windows. If not sure, new developers should default to Linux and the current long-term supported version of Ubuntu (Ubuntu LTS edition). If you are familiar with Docker, you can use one of the containers to install the development environment.

We use Debian / Ubuntu LTS as a standard supported version of Linux, and here my Linux version is Ubuntu 16.04.

3.1. Permission Setting

To facilitate development, we need to add users to the user group "dialout" to perform:

sudo usermod -a -G dialout $USER.

Then log out and log in again, because the changes are valid. Note: Never use sudo to fix permissions issues, or will cause more permissions issues that you need to be reinstalled to resolve.

3.2. Install the Dependency Package for PX4

Update the list of packages and install the dependency package for compiling PX4. Install the cmake first and perform:

sudo add-apt-repository ppa:george-edison55/cmake-3.x -y.

sudo apt-get update.

Install the necessary software, such as python, git, qtcreator, and so on.

3.3. Install the Cross-compilation Toolchain

The Ubuntu is equipped with a range of agent management that can seriously interfere with any robot-related serial port (or usb serial port), and uninstalling it will not matter, we can uninstall it here.Update the list of packages and install the following dependency packages.Install the arm-none-eabi compilation toolchain, and make sure to remove the residual before adding the arm-none-eabi toolchain.

3.4. Code Compilation

Once the development environment is built, you can start to download the code and compile it, and the PX4 can be developed in the console or in the graphical interface / IDE. Here we give a brief introduction to the console development.

Note that "make" is a character command compilation tool, "px4fmu-v2" is the hardware / ardupilot version, "default" is the default configuration, and all PX4 compilation targets follow this rule. By adding the 'upload' back to the command, the compiled binary program is uploaded to the flight-control hardware via the USB.

Of course, the compiled code in this way can not directly make the aircraft fly. If you want to download the code that can make your aircraft fly, it is recommended to download the QGroundControl ground station software, follow the software prompts to choose your own model, correct the sensor, link the remote control, debugging parameters and other operations.

4. PX4 Software Architecture Analysis

The software architecture of the PX4 will be stated from three aspects: the division of the software framework, the code file structure, and the flight control process. The source code analysis will select the representative return control module, hybrid controller and uORB middleware for analysis, and these three parts are subordinate to the application framework, library and operating system in the software framework hierarchy.

4.1. Division of the Software Framework

The PX4 software is built on the Nuttx operating system, and its general framework is shown in Figure 2:

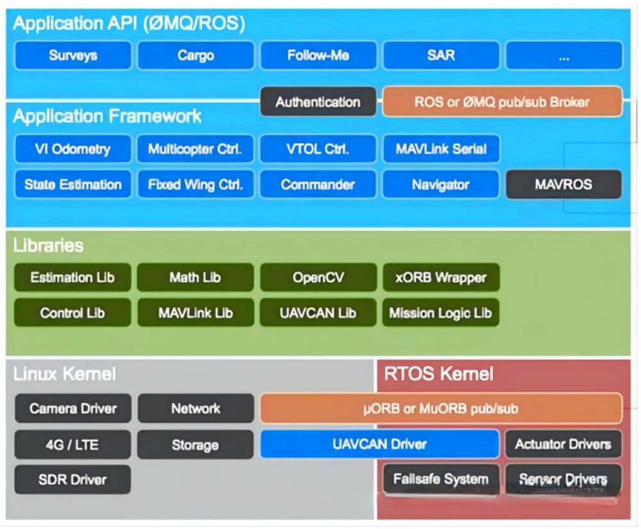

Figure 2. general framework

Depending on how the PX4 software works, four levels can be divided for a better understanding:

Application API: this interface to developers, such as using ROS (real-time operating system) or DroneAPI (DroneAPI is a Python library for control drones, can run separately on the airborne computer or other devices, through serial or wireless communication with flight control board), this layer is designed to be as far as possible, flat and hide its complexity.

Application Framework: This is the default assembly (node) for operating the underlying flight control. Such as rotor control, helicopter control, fixed wing control, position estimation, command control, navigation control and so on.

Library: This layer includes all the system libraries and basic traffic control functions. Like the math library, the control library, the MAV library, and so on.

4) Operating system: The last layer provides operating system support, such as task scheduling, hardware drivers, network, UAVCAN, and fault security system functions.

4.2. File Structure

After analyzing the overall framework, the analysis of the source code file structure is equally important, and the directory structure of the source code is shown in Figure 3.

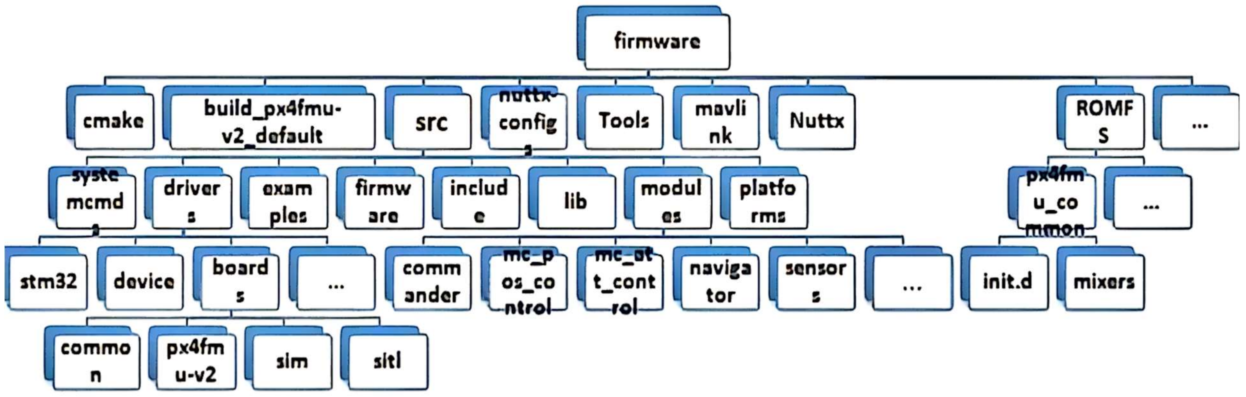

Figure 3. File Structure

There are many folders under the firmware folder, and we explain the functionality of the code in each folder in the order in the picture. Where cmake is the compilation tool; build_px4fmu-v2_default is the intermediate static file and final download file; src contains the main code for flight control; nuttx-config is the profile of nuttx; under tools folder contains tools such as download tools; Nuttx is the source code for Nuttx operating system; ROMFS contains the startup code of flight controller.

The following describes what is in the very important src folder. systemcmds: Mainly system tools that can be started or called in nsh. These include the control I2C, View the modified parameters, View the software version, Calibrate electrical adjustment, view system performance, bootloader upgrade and other tools; The drivers contains mainly the hardware-related driver code, For example, below the stm32 folder, it includes adc base classes, high-precision timers, servo-controlled drivers, External base class definition is included below the device folder, For example, I2C and SPI, etc., The boards directory defines the interface configuration of this type of board and the corresponding configuration interface (LED, PWM, USB, timer, etc.); examples contains some simple instance programs, If there is some experimental code to put under this; The include and lib directories contain headers and libraries for other code; System interface convenience and nuttx operating system separation are defined below platforms, This facilitates porting to other platforms; There are so many folders under the modules, These folders are each of the different modules, This is the top-most functional modules including commder, navigator, mc_att_control, mc_pos_control, etc.

4.3. Flight Control Process

The flow of flight control is shown in Figure 4. On the left side of the figure is the control function implementation of the flight control system, which we analyze from top to bottom and from left to right.

(1) The user controls the aircraft through the ground station or the remote control issuing mode switch and the rocker operation. The commander determines the state main_state that the user wants to switch to according to the current state of the aircraft: whether it can switch to the target state and determine the final state.stickmapper see name, joystick mapping, see px4io.c, sensors.cpp document.

(2) The navigator determines what the aircraft will eventually do with the conversion of the flight mode, nav_state, and the flight control of the aircraft in the corresponding mode.

(3) Position control to extract local postion (optical flow + IMU) or global postion (GPS) data for cascade PID control according to the navigation state. Outer ring input position difference output speed, inner ring output thrust.

(4) The attitude control adopts the uncoupling control mode of tilt separation, first align the Z axis and then the yaw Angle. The same cascade PID control mode is adopted, the outer ring input attitude angle error, the output attitude angle rate, the inner ring final output of the required torque.

(5) The mixer hybrid controller distributes the torque according to the model. According to the chlorophyll theory, the s-lift / torque generated by the propeller rotation is correlated to the square of the rotation speed, as well as the pull coefficient / torque coefficient. Construct a moment distribution matrix.

(6) The motor_driver drive motor to control the aircraft flight.

On the left is the implementation of the position estimation algorithm, where GPS determines global postion and optical flow determines local postion, and one of them can enter the POSCTL mode. Attitude estimation is the basis of everything, by fusing the advantages of inertial sensors (and optical flow) to calculate the current attitude information of the aircraft.

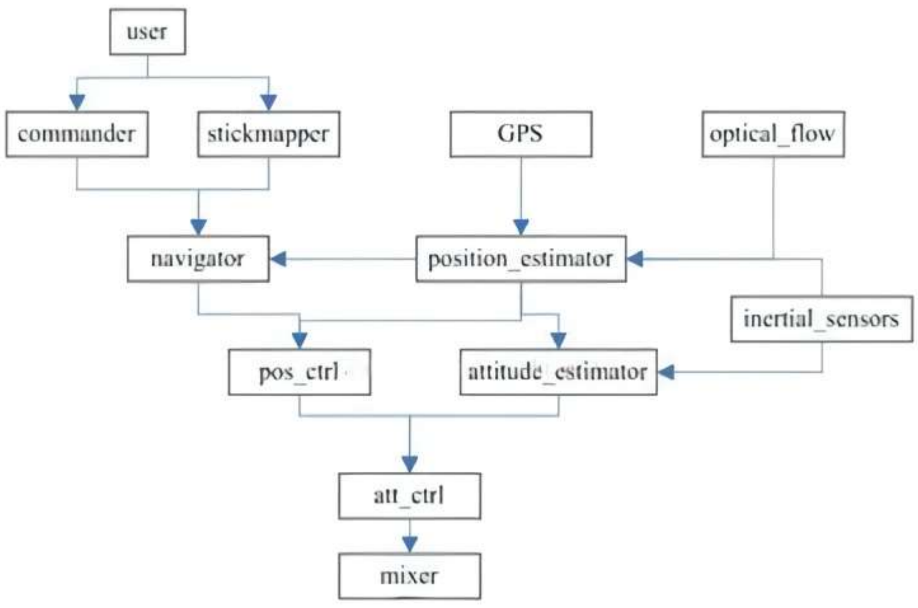

Figure 4. Flight control flow

5. Conclusion

The function of Pixhawk hardware has been able to meet the hardware needs of most aircraft, and due to the characteristics of open source hardware, it is believed that the hardware functions will be more and more powerful in the future. The PX4 software code is built on the Nuttx operating system, and the stability can be greatly guaranteed, while the PX4 modular programming idea also provides great convenience for developers to read and develop secondary things.

References

[1] Research on Obstacle Avoidance Technology of Multi-rotor UAV based on PIXHAWK Open source Flight Control [J]. Wang Tingling, Lu Duyang, Ma Yue-tao. Microcontroller and Embedded system Application. 2017(10).

[2] Yuan Dong, Chen Xufang, Yu Lele. Low Temperature Building Technology. 2017(02).

[3] Design of quadrotor manipulator UAV based on PIXHAWK Flight control [J]. Hu Yuzhu, XIAN Junheng, REN Shuping, DUAN Yueting, Li Yongpei. Electronic World. 2020(07).

[4] Research and Design of High Reliability Flight Control Based on PIXHAWK Architecture [J]. Wu Qi-qi, Li Shu-yuan, Huang Qing-nan. Industrial Control Computer. 2019(11).