Navio - Autopilot for Raspberry Pi

There is no need to explain all of the features of Raspberry Pi and how it is making  embedded development easier. However autopilot applications were previously limited on the platform.This is why we started Navio project.

We worked for a long time on a commercial UAV autopilot, which was based on STM32 microcontroller - the same microcontroller, that is used in most of the known autopilots. And when most of development of the core software was done our everyday job became integration of new payloads, new means of communication and new sensors.  And this process was very complicated, we had to write drivers for each new piece of electronics, even though we knew that they already exist. Development and debugging tools were bulky and kept us tied to the workplace. This is when the idea of Linux autopilot first crossed our minds. At first Navio started as a hobby project, that we were working on weekends. And each day our confidence grew, till the day we decided to quit our jobs to work on the Linux autopilot full time.

So we started investigating in what is the best way to create open and powerful Linux autopilot platform, and Raspberry Pi was an obvious choice. Great support from the community meant a lot of readily available code and tutorials. And we had all of the power and flexibility of Linux. When we first sent data over 3G network we just couldn't believe how easy it was, compared to what we had before. It was just like on a desktop PC. Simply install the drivers and you are ready to go!

But autopilot is much more than just a processor running OS, it needs a way to determine its state, its place on Earth, and be able to control actuators. And it takes many sensors and servos to fly a plane or multicopter. We needed an 9DOF inertial measurement unit for orientation, GPS for position, barometric pressure sensor for altitude and multiple control outputs. We got to the drawing board, well technically not to a drawing board, but to a CAD software, and designed the Navio. In the process an ADC was added for voltage and current monitoring, RGB LED to show statuses and connectors were made compatible with existing hardware.

Navio sits on top of the Raspberry Pi GPIO header and is fixed with standoff and bolt. It is very compact, we tried to make it as small as possible to fit in tight spaces inside small drones, but it packs everything needed for a full featured autopilot. For our own plane we have chosen to remove some of the sockets on Raspberry Pi to save weight and reduce size.

Here is a detailed list of sensors:

MPU9250 - 9DOF IMU chip, it has 3-axis accelerometer, 3-axis gyroscope and 3-axis magnetometer

MS5611 - a barometric pressure sensor with 10cm resolution.

u-blox NEO - GPS/Glonass receiver.

ADS1115 - 4-chanel 16bit ADC

PCA9685 - PWM generator. Controls up to 13 servos or motors and RGB LED.

When we first presented the idea to the community, the feedback was great. Raspberry Pi autopilot actually sounds sweet. But many had doubts, the autopilot has to be strict real-time, and it is not something you expect from Linux straight out of the box. So we started experimenting. From the previous experience with low latency audio processing on Linux, we knew that it can work real-time if tuned the right way.

With real-time kernel and proper task priority settings we have outperformed what we had on STM32. We were able to run IMU code at 3000 times per second, including reading data from sensors, computing roll, pitch and yaw angles and outputting them through wi-fi to console. Even when running alongside computational heavy processes like compiling, the performance was stable. This assured us that there is no need to overcomplicate the setup with secondary processor for sensor reading.

In our opinion, Raspberry Pi with Navio is not a commercial autopilot product, but something that lets you evaluate your ideas and test new algorithms. And the key factor here is ability to run multiple applications with autopilot in parallel. The other feature of Raspberry Pi is the ease of development, just open your favorite code editor, compile the file and run the application.

So hardware and OS were settled and we moved to autopilot code itself. At first we created small examples to check if the sensors and peripherals were working as expected, and as we got Navio to control servos it was time to try it on something real. We took an rc car off the shelf, installed Navio on top, added webcam and Wi-Fi . After several hours of coding we were able to control it from laptop keyboard and to get live video stream on the screen! Immediately we took a tour around the house, feeling like secret service ROV operators. We revealed some corners that has not been vacuumed for years, and even found the place where second socks go.

Then we needed software to actually fly a plane, but we did not want to reinvent the wheel. After working on one it was clear, that there is no quick way to write an autopilot. It takes a lot of testing and working on the details to actually make a reliable system. We all know how many open source autopilots are out there, it was just a question of picking the right one.

We were following ArduPilot project from the very beginning, and when we found out that the team is working on Linux port, we decided that we should make it run on our platform. Mostly this requires small changes to the hardware specific parts, which we are actively working on. The test platform is already assembled, and we are sure that we are going to take flight soon!