Pol. J. Appl. Sci., **2018**, *4*, 110-113

HIERARCHICAL ARCHITECTURE OF AN AUTONOMOUS UNMANNED AERIAL VEHICLE (UAV) USING MULTIWII AND RASPBERRY PI

Mateusz Andrzejewski, Adam Szmigielski

Department of Computer Science
Polish-Japanese Academy of Information Technology, Warsaw, Poland

E-mail: aszmigie@pjwstk.edu.pl

Abstract: The tasks performed by the autonomous robot are complex and multi-faceted. The robot should perform low-level tasks related to maintaining the set flight parameters as well as make complex decisions related to the robot task. It would be difficult to accomplish these tasks based on a simple control architecture. In many classic solutions, a hierarchical architecture is used, where the lower layers are responsible for the operation of sensors and actuators in accordance with the control algorithms. Higher layers are responsible for the tasks of navigation, location and implementation of set goals. In our solutions based on open hardware and software, we propose a robot architecture where low-level control is implemented by the Arduino controller with MultiWii software and higher tasks are carried out by the Raspberry Pi computer using the Linux operating system. The solution proposed is very flexible and allows quick and relatively easy prototyping. Another important advantage is the open nature of the hardware and software.

Key words: autonomous robots, unmanned aerial vehicles, quadrocopter, hierarchical control, open hardware

DOI: 10.34668/PJAS.2018.4.3.05

The construction of an autonomous UAV

To build an autonomous, intelligent robot, we have to solve many hardware and software problems. Very often the division between the software and the hardware is not obvious. An even more difficult case is presented with control algorithms, where the robot often has to perform many tasks at the same time, where these tasks depend on themselves as well as affect each other.

An example of this is the task of tracking an object. The robot must know how the object is described, must identify it, and then change its position, i.e. maintain the trajectory of the flight in accordance with the targeted objective. In the hardware layer, the UAV has many sensors such as orientation sensors (an acceleration sensor, a gyroscope, a compass), an altitude sensor (distance or atmospheric sensor), and a camera. Information from these sensors should be read and sent to the flight controller, whose task is to ensure stable UAV operation, e.g. keeping the set altitude, flight trajectory etc.

In complex tasks, related to the identification of objects, the determination of their mutual location or decision making, sensory information alone is insufficient. It must be subject to various types of filtration, extraction of features, and very often the object is identified based on these features. Decision-making processes also require searching through complex solution spaces. All these tasks require very large computational powers, whereas computers capable of performing such calculations are usually heavy and of considerable size. It is not without significance if we want to place them on relatively small size aircraft. To increase code clarity and reliability, these computers should be independent of low-level control tasks.

In our solution, we used two computers. One maintained flight control and its parameters, while the other ran and tested applications. In the case of a flight controller, we used the ATMega32u4 microcontroller with the Arduino bootloader, where we adopted the free MultiWii software. In the case of a computer for numerical calculations, we decided on the Raspberry Pi 3.

Hierarchical architecture of control

Low-level tasks are performed by the MultiWii flight controller, while tasks aimed at achieving the goal are performed by the Raspberry Pi computer. The block diagram of the UAV control is shown in Fig. 1.

In principle, the UAV presented is a fast, cheap and convenient platform for testing complex localization algorithms, object tracking algorithms and other intelligent algorithms. It is obvious that we are more focused on writing and testing algorithms than on building a UAV. To achieve this, we focused on closing the stage of building a UAV and organizing the environment for testing algorithms. In principle, when using a UAV, we should not concentrate on any interference with the hardware layer and the flight controller, but only on the tested algorithm. This is possible after completing the design work with the equipment and determining the communication protocol between the flight

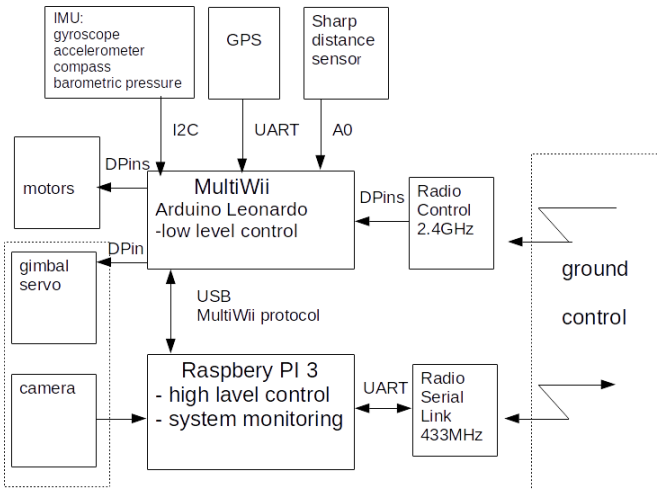

Fig. 1: Architecture of our system.

control and the host. We have adopted the MultiWii serial communication protocol [1].

Low-level control architecture

As mentioned earlier, we do not assume modifications of low-level UAV control during normal use. However, such modifications related to the inclusion of new elements are possible and do not require a lot of effort. In this part of the article, we will present the construction of the UAV hardware layer, the manner of their organization and the implementation of control tasks.

Body frame of UAV We used popular carbon fiber frame – QAV 250 – Mini Quadcopter FPV. The QAV 250 is a symmetric 250mm size airframe, measured motor shaft to motor shaft diagonally.

Sensors We used typical sensors necessary for the proper operation of the UAV.

Inertial Measurement Unit IMU GY-88 is 10 DOF a motion tracking module, that combines a 3-axis gyroscope and 3-axis accelerometer. It also has a 3-axis digital compass and a high-accuracy chip to detect barometric pressure and temperature.

Sharp distance sensor GP2Y0A710K0F – smart infrared sensor for distance measurement. We use it to stabilize altitude in heights of up to 5 m.

GPS NEO6MV2 – simple module for fixing GPS position.

Actuators – In our project we used a brushless motor.

motors – we used 4 brushless motor 2204 2300KV Motor with 12A controller.

Gimbal servo Tower Pro SG90 9g – to stabilize the camera a small servo is used.

Radio control For safety reasons, we used two independent wireless communication lines. The first is a 6-channel radio control system typically used for controlling flying models. A digital system with Automatic Frequency Hopping technology, using 2.4 GHz radio band, provides uninterrupted connectivity with a range of up to 1000 m. This radio can be used in parallel for autonomous work; it supports the launch of the UAV as its landing. In the absence of communication with the UAV, and when the UAV does not perform any autonomous tasks, the UAV automatically lands on the ground.

The second wireless line is used to communicate Raspberry Pi with a ground control station. We used module HC-12, with theoretical working distance up to 1000m. For safety reasons, the UAV can perform the algorithm autonomously if itis in range of the radio. In addition, it is possible to send short information about the failure or state of the UAV. In order not to interfere with the modeling radio, the serial communication module uses frequency 433MHz.

Low-level controller As a low-level flight controller we used a system containing a microcontroller controller ATMega32u4. The use of simplified microcontroller architecture ensured a high level of reliability and high speed execution of the code which translates directly into the safety of the drone moving in three-dimensional space. The main tasks performed by the low level controller are:

- stabilization of the UAV vehicle.
  The drone we are considering is aerodynamically unstable in both static and dynamic contexts. Its effective use is based on using fly by wire control and active stabilization technology that allow the drone to return to the equilibrium state due to the active reaction of the controller to external disturbing forces. Two stabilization modes have been implemented. The first is based on the use of the information of accelerometer sensor value change for each axis of the Cartesian coordinate system associated with the UAV. Due to the high responsiveness it is mainly used during a flight actively controlled by a human operator (acro mode). The second based on the use of a three-axis gyroscope ensures a high level of stabilization and is mainly used for autonomous UAV flights (angle mode). There is also an intermediate mode that is a linear assembly of the two above (for small control signals dominance angle mode and for large acro mode). To maintain the position in the Z axis (height) in the range 0-5m, we use the data from the optical position sensor (ultrasonic sensors have a limited field of application due to the interference generated by the propeller unit). In the range above 5 m we use data from the barometric sensor.
  The values of the observed physical values are approximated from the sensors with the help of a complementary filter, enabling the elimination of measurement errors of one sensor by integrating the indica-

tions of other sensors and taking into account previous measurements. For example, a gyro drift is minimized by taking into account data from the accelerometer.
The more effective Extended Kalman Filter (EKF) algorithm was rejected due to its computationally expensive cost and the limited improvement in the quality of results. To increase the accuracy of the sensor readings, the pre-start activation procedure includes the calibration of the accelerometer and barometric sensors.

- RC commands handling.
  A supported RC receiver transmits to the controller signals in the form of PWM (pulse-width modulation) corresponding to the rotation round each axis of the Cartesian coordinate system associated with the drone. The signal responsible for controlling the power of engines and additional signals used as function switches is implemented on the drone platform. Rotation control is carried out by maintaining relative speed differentiation for each engine pair. Due to the aerodynamic design, the deflection of the drone from the XY plane causes it to move in a certain direction.
- high level controller commands handling.
  Due to the UART interface connection with the higher-level controller, the controller receives commands which control the behavior of the UAV according to the planned algorithm. The connection is two directional; a lower level controller continuously sends information to the higher-level controller from its sensors , and control pulses are received using the RC receiver.
- emergency situation handling.
  An important feature of the lower level controller is the handling of emergency situations. If the defined parameters are exceeded for information from sensors or control systems, the drone can perform the planned action. For example, for a situation of loss of signal from the RC receiver (no connection), the drone makes a steady landing at a low descent speed. In the case of the version of the UAV equipped with GPS, it is possible to return to the home position before starting the landing procedure.

High-level control architecture

Raspberry Pi 3 has a powerful Broadcom processor with underlying architecture of the BCM2837 with a quad-core ARM Cortex A53 (ARMv8) cluster and 1 GB RAM memory. The ARM cores run at 1.2GHz [2]. That makes Raspberry Pi 3 comparable to a PC computer due to computing effectiveness, and it is much smaller due to weight, dimensions and power consumption. Consequently, it is an ideal computer unit to be applied in unmanned aerial vehicle as a control system. Additionally, Raspberry Pi 3 can run Linux with many programming libraries and languages. In our project we often use OpenCV, MultiWii libraries with Python language programming.

The Raspberry Pi 3 computer is connected directly to the camera. This enables real-time image analysis. Consequently, it is possible to use image-based control algorithms, object tracking, classification or decision making. In addition, the camera is placed on the gimbal, which is based on information from the IMU sensors (gyroscope and accelerometer), enables the camera to be kept in a fixed position, i.e. perpendicular to the earth's surface.

Tasks controlled by high-level controller. The tasks performed by Raspberry Pi are not strictly defined. They depend on what the UAV is used for. It is an open experimental field, and the presented system, in our opinion, is an interesting solution for testing the operation of new control algorithms. In general, it can be concluded that the tasks performed by the Raspberry controller are tasks that require some kind of intelligence. Such as:

- Object trucking;
- Relative localization;
- Goal-oriented tasks;
- Deep learning algorithms.

Autonomous Systems

To be able to talk about an autonomous robotic system, all systems (mechanical, electronic, sensory, algorithmic, etc.) should be integrated into an efficient whole. As mentioned earlier, Arduino MultiWii performs low-level tasks, and you can use intelligent algorithms with Raspberry Pi.

Fig. 1: Autonomous UAV.

For full system integration, a communication protocol between Raspberry Pi and Arduino is needed.

MultiWii Serial Protocol. The MultiWii project provides such a protocol. Depending on the needs, it can be modified. The general format of an MultiWii Serial Protocol message is:

$$< preamble >, < direction >, < size >, < command >, < crc >$$

where:

preamble = the ASCII characters '$M'
direction = the ASCII character '<' if to the MWC or '>' if from the MWC
size = number of data bytes, binary. Can be zero as in the case of a data request to the MWC
command = message id as per the table below
data = as per the table below. UINT16 values are LSB first
crc = XORof < size >,< command > and each data byte into a zero'ed sum

It should be noted that recently libraries for communication with MultiWii have been created and are constantly revised based on a defined protocol.

Using UAV and algorithm testing

Having an autonomous aircraft platform, we can start writing and running applications on it. In this part of the article we will present a few remarks regarding the writing and running of the program, monitoring the tasks performed and dealing with security issues.

Secure running application

Because Raspberry has a full Linux system installed, it is possible to choose any programming language and runtime environment. We decided on Python. The Raspberry Pi computer has a Wi-Fi module, so we can log in to Raspberry remotely and run any program. For safety reasons, in order to prevent any uncontrolled behavior of the UAV we assume that the UAV can behave autonomously as long as it is within the range of the RS wireless connection. The theoretical range of this connection, declared by the manufacturer, is 1000m. In the absence of communication Raspberry Pi does not take any control. Then we can use manual radio control from the MultiWii level. In the absence of communication and with this radio, the UAV launches the automatic landing procedure (failsafe).

UAV state monitoring

In the case of checking the correctness of the algorithm's operation, the monitoring of flight parameters, camera view, etc. we decided to save the parameters that interested us on the SD card. The analysis of the algorithm's operation is done offline. Image streaming while the algorithm is running, can significantly load the processor; it requires an additional wireless video link that can potentially introduce interference, especially in the 2.4 GHz band and additionally it loads the UAV and consumes power.

Conclusion and further work

In this work, we presented a proposal to build a UAV platform using hierarchical control architecture. For the construction of the UAV, we used both software and hardware generally available under the terms of free licenses. This allows you to quickly, cheaply and efficiently build a UAV that can be used for more complex tasks.

We are currently working on tracking algorithms using deep learning techniques and locating and navigating a larger group of UAVs.

Literature

[1] MultiWii Serial Protocol http://www.multiwii.com/wiki/index.php?title=Multiwii_Serial_Protocol.
[2] Raspbery PI https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

Received: 2018
Accepted: 2018