Work-in-Progress Abstract: Revealing and Analyzing Architectural Models in Open-source ArduPilot

Sergey Staroletov
Polzunov Altai State Technical University / Institute of Automation and Electrometry
Siberia, Russian Federation
serg_soft@mail.ru

***Abstract*—Building robust software can be considered a major challenge in current software engineering processes. This task is especially relevant for the code of cyber-physical systems (CPS) that interact with tangible data of the environment and make decisions that have an impact on the real world. The study of good practices of the architectural organization of such software systems is suitable to conduct on solutions with open-source code, which are developed by large communities of enthusiasts. Such a code bears a long history and has been tested many times on real devices in a real-world environment. The construction of various models using the program code allows us to understand stable architectural solutions, to present them in a graphical form; these solutions can be used in STEM centers when designing other systems, taking into account all the achievements of the communities. In addition, it is possible to propose methods for analyzing models to prove various properties of cyber-physical systems.**

In this paper, we analyze ArduPilot Mega (APM), an Arduino-compatible solution for building DIY driving and flying systems. The solution is based on a specially designed board with a controller and necessary peripherals, as well as a firmware code in a C++-compatible dialect. Since there are many limitations associated with hardware, it is advisable to carry out a so-called co-modeling, taking into account both hardware and software sides. We consider modeling the interaction of equipment on connected pins and data transmission buses, the software part in the form of a class diagram for the solution. We then describe methods for analyzing the interactions between tasks running on the system through shared variables and evaluating the performance of the task scheduler.

I. Introduction

While Edward Lee defines cyber-physical systems (CPS) as integrations of computation with physical processes [1], we can assume that the system part is significant and needs to be analyzed. This part can be represented by a real-time operating system or firmware. One interesting example in the domain of CPS is quadcopters (or quadrotors [2]), they are now successfully used for videography, weather measurement, or monitoring agricultural lands [3]. Such systems are also interesting for describing and proving properties using model checking [4] and theorem proving methods [5]. In this work, we consider small DIY quadcopters for amateurs [6]. To understand practices (or patterns) of developing software for such devices, we are studying the ArduPilot/ArduCopter project [7] and its original branch [8] for Arduino devices. The hardware controller of such a branch is cost- and energy-efficient, at the same time, its software is optimized as much as possible to take into account the hardware restrictions of the platform and the requirements for the size of the code and the memory used. In modeling, we mainly use the AADL language (described in [9]), which is widely utilized for modeling reliable systems, and OSATE tool [10]. This language allows engineers to decompose the system in a simple language of property sets and interactions between components, as well as merge models in other modeling languages into a single model using Annexes [11]. We also apply UML and automata models.

II. Modeling the hardware part

The ArduPilot Mega board mainly comprises: (1) ATmega 2560 processor (8bit) [12]; (2) barometer MS5611 [13], SPI connection; (3) 3-axes magnetometer (compass) HMC5843 [14], I2C connection; (4) 6-axes gyroscope and accelerometer MPU-6000, SPI connection; (5) GPS, UART external interface; (6) Frsky telemetry [15], UART external interface; (7) an additional external SPI interface.

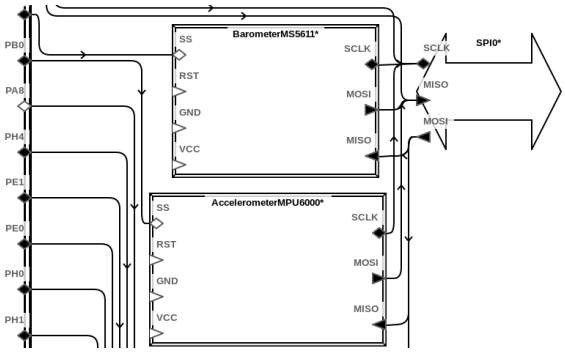

Fig. 1. A fragment of the APM board model, built from our AADL code

By modeling hardware, we here mean the connections between various components of the board via pins, as well as the use of data buses. For such modeling, we carefully walk through the source code and study the communications to the GPIO ports of the ATmega processor. As a result, we offer Fig. 1 as part of a schematic that contains components (processor, peripherals, buses) and connections between them, described in AADL and represented graphically. The current state of modeling is posted on GitHub [16]. Our models more

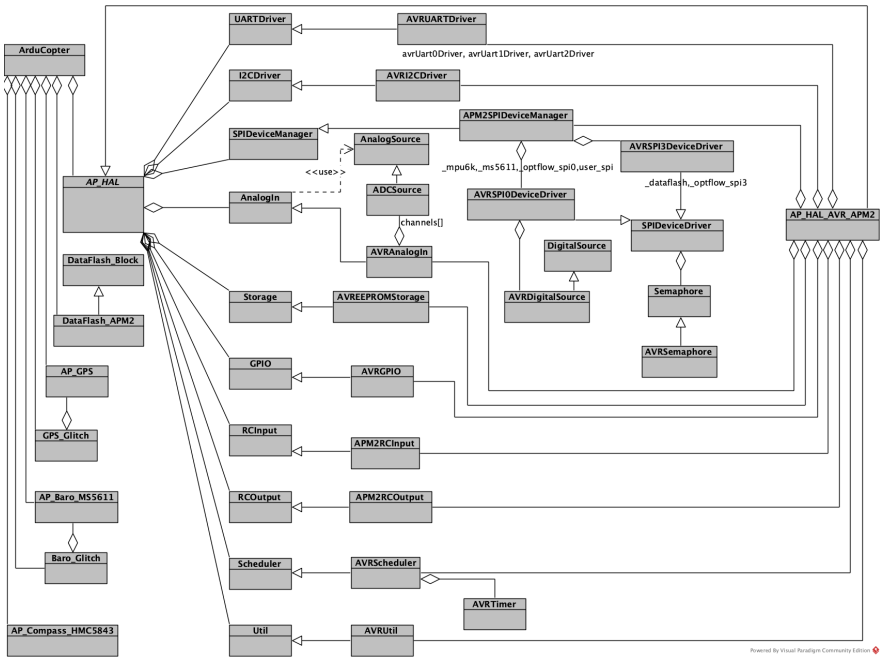

Fig. 2. Some notable classes and their relations in the implementation of ArduPilot/ArduCopter

accurately represent the actual hardware than some existing open-source works [17], [18]. We are also going to take into account the electrical connections between components.

III. Modeling the software part

First, we would like to represent the relationship between the current classes as a class diagram (Fig. 2). Here one can see the architecture of the solution to support different possible types of devices, such as GPS and barometer. Secondly, we are interested in the logical processes operating in this system and actually providing the possibility of flight and control. We have identified such processes in the source code as scheduler tasks (Fig. 3). Currently, we are implementing software for automatic source-code analysis of connections between such tasks through shared variables. We analyze forests of control-flow graphs [19] with functions and monitor read and write access to variables. We expect to obtain relations between our 30+ tasks and generate an AADL code of the form [20].

IV. Modeling the scheduler

Scheduling in ArduPilot 3 refers to non-preemptive periodic tasks. Each task (Fig. 3) is characterized by its period and the maximum estimated operating time. When choosing a task, the scheduler should take into account its interval and select those tasks that can still be completed by their maximum duration in a given time window (the scheduler period is one second). The remaining time is also taken into account. An interesting challenge is to check the correctness of this scheduling. Here we can propose a modification of our approach [21] with the implementation of a non-preemptive scheduler in Promela [22]. We can determine our tasks that are virtually executed for some time less than the specified maximum, implement

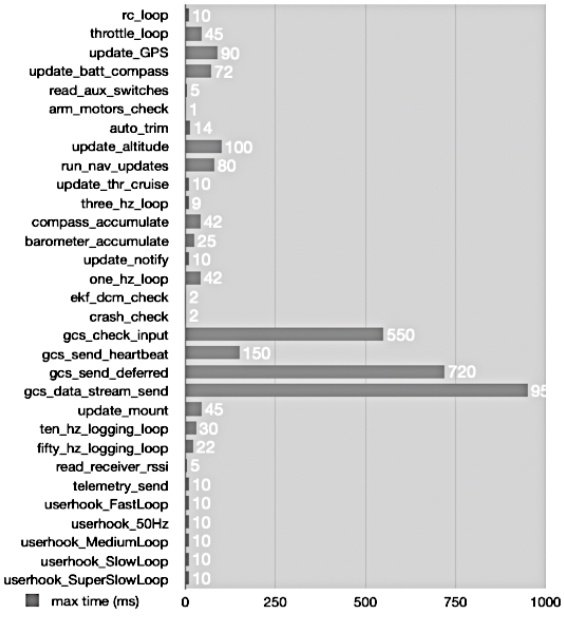

Fig. 3. Tasks in ArduCopter and their estimated execution times [23]

the algorithm for their scheduling according to the ideas in the original code and check that a particular task gets its time and investigate the reasons for potential non-executions.

V. Conclusion

In this paper, we considered approaches to modeling software for the controller of a complex cyber-physical system. Further, these models can be used to analyze things like schedulability, reliability in case of failure of various components, or analysis of delays by transferring sensor data over the data buses.

REFERENCES

[1] E. A. Lee, "Cyber physical systems: Design challenges," in *2008 11th IEEE international symposium on object and component-oriented real-time distributed computing (ISORC)*. IEEE, 2008, pp. 363–369.

[2] Q. Quan, *Introduction to multicopter design and control*. Springer, 2017.

[3] K. R. Krishna, *Agricultural drones: a peaceful pursuit*. CRC Press, Tailor & Francis group, 2018.

[4] S. Staroletov and N. Shilov, "Applying model checking approach with floating point arithmetic for verification of air collision avoidance maneuver hybrid model," in *International Symposium on Model Checking Software*. Springer, 2019, pp. 193–207.

[5] S. Staroletov, "Automatic proving of stability of the cyber-physical systems in the sense of Lyapunov with KeYmaera," in *2021 28th Conference of Open Innovations Association (FRUCT)*. IEEE, 2021, pp. 431–438.

[6] C.-H. Lai and C.-M. Chu, "Development and evaluation of STEM based instructional design: An example of quadcopter course," in *International Symposium on Emerging Technologies for Education*. Springer, 2016, pp. 176–191.

[7] *ArduPilot Copter Project*, 2020. [Online]. Available: https://ardupilot.org/copter/

[8] *ArduPilot Project*, 2015. [Online]. Available: https://github.com/ArduPilot/ardupilot/tree/ArduCopter-3.2.1

[9] J. Delange, "AADL in practice: Become an expert in software architecture modeling and analysis," *Reblochon Development Company*, 2017.

[10] P. Feiler, "The open source AADL tool environment (OSATE)," Carnegie Mellon University Software Engineering Institute Pittsburgh, Tech. Rep., 2019.

[11] J. Delange and P. Feiler, "Architecture fault modeling with the AADL error-model annex," in *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*. IEEE, 2014, pp. 361–368.

[12] *Atmel ATmega640 / V-1280 / V-1281 / V-2560 / V-2561 Datasheet*, 2014. [Online]. Available: https://www.microchip.com/wwwproducts/en/ATmega2561

[13] *MS5611-01BA03 Barometric Pressure Sensor, with stainless steel cap*. [Online]. Available: https://www.te.com/commerce/DocumentDelivery/DDEController?Action=showdoc&DocId=Data+Sheet%7FMS5611-01BA03%7FB3%7Fpdf%7FEnglish%7FENG_DS_MS5611-01BA03_B3.pdf

[14] *Honeywell 3-Axis Digital Compass IC HMC5843*. [Online]. Available: https://www.sparkfun.com/datasheets/Sensors/Magneto/HMC5843.pdf

[15] *FrSky Electronic*. [Online]. Available: https://www.frsky-rc.com

[16] *Co-modeling Ardupilot*, 2021. [Online]. Available: https://github.com/SergeyStaroletov/Co-modeling-Ardupilot

[17] J. Hugues, "AADLib, a library of reusable AADL models," SAE Technical Paper, Tech. Rep., 2013.

[18] *OpenAADL/AADLib boards-ardupilot*, 2018. [Online]. Available: https://github.com/OpenAADL/AADLib/blob/master/src/aadl/boards/boards-ardupilot.aadl

[19] L. Serrano, "Automatic inference of system software transformation rules from examples," Ph.D. dissertation, Sorbonne Université, 2020.

[20] *POK Ardupilot example*. [Online]. Available: https://github.com/pok-kernel/pok/tree/main/examples/case-study-ardupilot

[21] S. M. Staroletov, "A formal model of a partitioned real-time operating system in Promela," *Proceedings of the Institute for System Programming of the RAS*, vol. 32, no. 6, pp. 49–66, 2020.

[22] V. Natarajan and G. J. Holzmann, "Outline for an operational semantics of Promela." *The Spin Verification System*, vol. 32, pp. 133–152, 1996.

[23] *ArduPilot/ArduCopter Project. Tasks*, 2015. [Online]. Available: https://github.com/ArduPilot/ardupilot/blob/ArduCopter-3.2.1/ArduCopter/ArduCopter.pde#L777