Instrumentation of Embedded Software, Use Case: Drone Autopilot

Master's Thesis

Supervisor: Prof. Dr. Alexander Pretschner
Advisor: Ehsan Zibaei
Email: {pretschn, zibaei}@in.tum.de
Phone: +49 (89) 289 - 17885
Starting date: immediately

Fakultät für Informatik
Lehrstuhl 4
Software & Systems Engineering
Prof. Dr. Alexander Pretschner

Boltzmannstraße 3
85748 Garching bei München

Tel: +49 (89) 289 - 17885
https://www4.in.tum.de

Context

Drones are becoming increasingly popular because of their unique capabilities in transportation and monitoring, however they are not still safe enough to be integrated into the public air space. One way to improve a system's safety is to develop detective mechanisms to learn from previous (mis)behaviors and increase the accountability of the system. A system can be made more accountable by inclusion of a detective mechanism to log necessary information during the operation and analyze them post-mortem[1]. Such a diagnosis tool can be very beneficial for the drone stakeholders. For example, users can have a better understanding of the root cause of their crash and try to avoid their mistake in the next flights. designers can learn from previous faults and improve the safety of their design. Testers can propose more effective test cases by analyzing the previous failures.

Richness of the logs have significant effect on the outcome of the diagnosis process. Instrumentation applied at run-time is called dynamic instrumentation and instrumentation applied at compile time is called static instrumentation[2]. Our available drone autopilot[3], has only a static instrumentation mechanism to profile the main loop execution time. The generated information by that mechanism is not rich enough for diagnosis applications. Thus we need to instrument the autopilot software to generate more information about the autopilot runtime behavior. There are several available tools in the literature for instrumenting an embedded software [4,5,6,7] but there is no concrete evaluation on complex embedded systems such as drones. In this project, we will survey the state of the art in instrumenting time-critical software and compare the applicability of available technologies for instrumenting our drone's autopilot.

Goal

The goal of this thesis is to instrument and analyse an autopilot software for drones, while considering the granularity of the logs and the performance of the autopilot

Working Plan

1. Do a survey on static and dynamic embedded software instrumentation
   (a) What kind of information can be generated by instrumenting a software (tracing, profiling and etc.)
   (b) What are the limitations in instrumenting an embedded software
   (c) Write a state-of-art survey of instrumentation technologies for embedded systems
2. Analyze the autopilot software
   (a) Identify main components of the autopilot software and its architecture
   (b) Identify performance limitations of the autopilot available in the lab
   (c) Analyze applicability of existing instrumentation mechanisms for logging the autopilots
3. Instrument the autopilot code
   (a) Implement one static instrumentation mechanism on the autopilot
   (b) Implement one dynamic instrumentation mechanism on the autopilot
   (c) Implement one parallel hardware instrumenter mechanism on the autopilot

4. Analyze the results
   (a) Analyze the performance of the instrumented software on the simulation environment and the real drone
   (b) Discuss the trade-offs between logging granularity and performance of the autopilot
5. The final thesis document must contain:
   (a) Description of the problem and motivation for the chosen approach
   (b) State of the art survey of the literature
   (c) Rationale for choosing certain technique(s) for implementation
   (d) Implementation description
   (e) Performance evaluation of different instrumentation techniques
   (f) Discussion on potential performance issues
   (g) Conclusions and future work

Deliverables

- Docker container with the source code of the implementation
- A demo of the implementation, including instructions on how to run the demo
- Technical report with comprehensive documentation of the implementation, i.e. design decision, architecture description, API description and usage instructions
- Final thesis report written in conformance with TUM guidelines

References

[1] Beckers, Kristian, Jörg Landthaler, Florian Matthes, Alexander Pretschner, and Bernhard Waltl. "Data accountability in socio-technical systems." In Enterprise, Business-Process and Information Systems Modeling, pp. 335-348. Springer, Cham, 2016.

[2] Sun, Enqiang, and David Kaeli. "A binary instrumentation tool for the Blackfin processor." In Proceedings of the Workshop on Binary Instrumentation and Applications, pp. 43-51. ACM, 2009.

[3] ArduPilot Open Source Autopilot. Ardupilot.org. Retrieved from `http://ardupilot.org/`

[4] Nethercote, Nicholas, and Julian Seward. "Valgrind: a framework for heavyweight dynamic binary instrumentation." In ACM Sigplan notices, vol. 42, no. 6, pp. 89-100. ACM, 2007.

[5] Kashif, Hany, Pansy Arafa, and Sebastian Fischmeister. "INSTEP: A static instrumentation framework for preserving extra-functional properties." In Embedded and Real-Time Computing Systems and Applications (RTCSA), 2013 IEEE 19th International Conference on, pp. 257-266. IEEE, 2013.

[6] Schardl, Tao B., Tyler Denniston, Damon Doucet, Bradley C. Kuszmaul, I. Lee, and Charles E. Leiserson. "The CSI Framework for Compiler-Inserted Program Instrumentation." Proceedings of the ACM on Measurement and Analysis of Computing Systems 1, no. 2 (2017): 43.

[7] Nair, Ajay, and Roman Lysecky. "Non-intrusive dynamic application profiler for detailed loop execution characterization." In Proceedings of the 2008 international conference on Compilers, architectures and synthesis for embedded systems, pp. 23-30. ACM, 2008.

Fakultät für Informatik
Lehrstuhl 4
Software & Systems Engineering
Prof. Dr. Alexander Pretschner

Boltzmannstraße 3
85748 Garching bei München

Tel: +49 (89) 289 - 17885
https://www4.in.tum.de