Toward a Java Based Infrastructure for Unmanned Aerial Vehicles

Yu David Liu
SUNY Binghamton, USA
davidl@cs.binghamton.edu

Lukasz Ziarek
SUNY Buffalo, USA
lziarek@buffalo.edu

Abstract

Unmanned Aerial Vehicles (UAVs) have recently emerged as a promising platform for civilian tasks and public interests, such as merchandise delivery, traffic control, news reporting, natural disaster management, mobile social networks, and Internet connectivity in third-world countries. Looking forward, the exciting potential of UAVs is accompanied with significant hurdles that call for broad and concerted interdisciplinary research, with diverse focuses on real-time system design, energy efficiency, safety and security, programmability, robotics and mechanical design, among others. This poster proposes an open-source and extensible software infrastructure for UAVs.

Categories and Subject Descriptors D.4.7 [*Organization and Design*]: Real-time systems and embedded systems; D.3.3 [*Language Constructs and Features*]: Frameworks

Keywords Unmanned aerial vehicles, software infrastructure, extensibility, Java

1. Introduction

In our opinion, an *open-source*, *extensible*, and *general* infrastructure is critical for promoting research and education in UAV systems, ultimately leading to a better understanding, assessment, and possibly wider application of this emerging technology. Prior work has focused on very low level embedded software / hardware platforms. To the best of our knowledge, only few projects — the most notable one is perhaps Paparazzi [5] — share our vision of developing an open infrastructure that a broader community of users with diverse interests can benefit. The most fundamental problem of Paparazzi — and similar systems such as AutoPilot [1] and OpenPilot [4] — is that UAVs are viewed as a hardware-centric embedded/robotics system. By openness and extensibility, those projects focus on offering refined support to interface with diverse models of autopilot boards, sensors, motion controllers, and so on. As a result, Paparazzi directly operates on micro-controllers with no operating system support; programming in Paparazzi is restricted to low-level embedded system programming; the support of high-level Application Programming Interfaces (APIs) is minimal; and advanced support for software quality (such as program analysis or verification) is non-existent. Overall, we believe a Paparazzi-like infrastructure may be suitable for a mechanical engineering researcher to prototype, or for a UAV aficionado to navigate model planes, but its standpoint may be too low-level to accommodate a broad spectrum of computer science research, and introducing UAVs in the classroom.

SPLASH Companion'15, October 25–30, 2015, Pittsburgh, PA, USA
ACM. 978-1-4503-3722-9/15/10
http://dx.doi.org/10.1145/2814189.2817276

2. jUAV Software Infrastructure

We propose an extensible software infrastructure for UAVs and their payload applications. We envision an infrastructure that enables the UAV as a unified computation platform: the developers can engage in a wide range of application tasks — data collection, monitoring and surveillance, physical object transfer, communication intermediary — and researchers can explore and meet a diverse set of requirements such as predictability, energy efficiency, program correctness, and application security. To support this vision, we introduce the following design goals:

Design Goal I [Whole-Stack Extensibility]: to promote UAV research from different areas, the new infrastructure should be structured in an extensible fashion so that researchers from different research areas can all effectively contribute.

Design Goal II [Portability]: since UAV hardware components are extremely diverse, the new system should encourage solutions portable to different architectures, operating systems, virtual machines, and compilers.

Design Goal III [Resource Awareness]: the new system must provide infrastructure-level support to account for system resources critical for UAV systems, such as energy, and expose them to higher levels of compute stacks to enable

and facilitate whole-stack research on addressing these constraints.

Design Goal IV [Software Quality]: UAVs in many applications are mission-critical. The infrastructure-level support should promote safety, and provide strong support for debugging.

Design Goal V [Payload Application Friendliness]: the new system must provide well-defined, intuitive, and high-level programming interfaces to accommodate diverse application domains for UAV software payloads.

Design Goal VI [Education Effectiveness]: the new system must be easy to understand, learn, and experiment with.

No existing UAV infrastructures we know of satisfy all of the above **Design Goals**. At its heart, our infrastructure is a software ecosystem that draws on existing technologies — including hardware-centric frameworks such as Paparazzi, real-time operating systems (RTOS), C-based compilers and linkers, real-time Java and its virtual machines — and integrate them in novel ways to achieve our listed **Design Goals**, while providing a useful infrastructure for enabling the research and educational for the broader community. It is important to point out that our proposed infrastructure is not meant to be a *universal* solution for all future UAV developments. Our intended scope of UAV support is "middleweight" — not too powerful like a "flying desktop," but not too resourceless like micro-UAVs [3, 6] with memory size in the KBs (*e.g.*, Robobees [2]). Instead, we will target commodity embedded boards.

2.1 Proposed Infrastructure

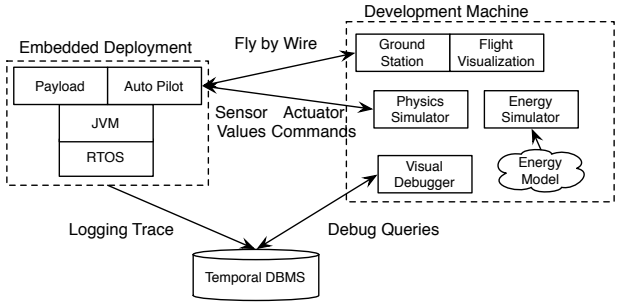

Figure 1: High-Level System Stack

Our proposed infrastructure is illustrated in Fig. 1. It consists of three distinct components, which can be deployed on the same machine for simulation, or separate machines in more realistic deployment environments: (1) an autopilot and software payload execution framework, typically executed on an embedded board; (2) a ground station and simulation support, typically executed on a desktop or development machine; and (3) debugging and tracing support.

To promote *Whole-Stack Extensibility*, we propose a UAV software infrastructure with a conceptually layered design including hardware drivers, (real-time) operating systems (OS), virtual machines (VM), and compilers. One attractive consequence of the design is *Portability*, where the diversity of hardware can remain abstract at OS, VM, or compiler levels, and the diversity of system software can remain abstract at the application level, promoting the platform independence of UAV payload applications. The layers in our design are *conceptual*: we offer flexible modes of compilation and interpretation support to maximize efficiency, so that *e.g.*, the VM and the OS can be compiled to a binary image with aggressive cross-layer optimizations, or a VM with direct support of load-time or run-time just-in-time compilation.

Java, as well as other languages that leverage the managed runtimes, are widely used for application-level programming. We propose to offer UAV programmers a Java programming interface, promoting *Payload Application Friendliness*. For instance, numerous Android Apps address tasks useful for UAVs, such as geo-based tracking, video/camera management, and motion detection and pattern recognition. With Java support, such Apps may undergo minimal or no modifications to serve as off-the-shelf building blocks for developing UAV applications. Another distinct advantage of the Java-based programming model is its strength in safety: Java guarantees type safety and memory temporal and spatial safety at the language level, with significant research and tool support on debugging, error handling, program repair, and verification. These technologies promote *Software Quality*. To accommodate *Whole-Stack Extensibility*, where some low-level programming is necessary at the hardware abstraction layer or the OS layer, our proposed framework also supports the C programming model with full interoperability and backward compatibility.

Acknowledgments

We thank the anonymous reviewers for the useful comments and suggestions. This work is sponsored by US NSF CNS-1512992 and CNS-1513006.

References

[1] Autopilot: Do it yourself uav, `http://autopilot.sourceforge.net`.

[2] Bryan Kate, Jason Waterman, Karthik Dantu, and Matt Welsh. Simbeeotic: A simulator and testbed for micro-aerial vehicle swarm experiments. In *Proceedings of the 11th International Conference on Information Processing in Sensor Networks*, IPSN '12, pages 49–60, New York, NY, USA, 2012. ACM.

[3] Gregory Mone. Rise of the swarm. *Commun. ACM*, 56(3):16–17, March 2013.

[4] Openpilot, `http://www.openpilot.org/`.

[5] Paparazzi: The free autopilot, `http://wiki.paparazziuav.org/`.

[6] Rj Wood, B Finio, M Karpelson, K Ma, No Pérez-Arancibia, Ps Sreetharan, H Tanaka, and Jp Whitney. Progress on 'pico' air vehicles. *Int. J. Rob. Res.*, 31(11):1292–1302, September 2012.