Understanding the Software and Hardware Stacks of a General-Purpose Cognitive Drone

Sam Jijina, Adriana Amyette, Nima Shoghi, Ramyad Hadidi, Hyesoon Kim
Georgia Institute of Technology

Motivation
- Commercial drone industry will reach 805,000 in sales in 2021, a CAGR of 51% [1]
- Increasing use of drones from surveying land to emergency services and national security
- Open-source flight stack to promote innovation through collaboration
- Characterizing underlying architecture and flight stack to achieve high reliability, safety, and performance

Applications of Drone Technology
- Aerial photography
- Agriculture
- Defense
- Emergency services
- Geographic mapping
- Personal hobby
- Search and rescue
- Shipping
- and many more...

Current Technological Shortcomings
- Most drone flight stacks are not open-source
  - Many popular models are closed due to market forces
- No access to the autopilot code base
  - Some manufacturers are using custom chips
- Weight carrying capacity limitations
- General purpose drones not designed to carry large payloads
- Very difficult to alter hardware due to custom PCBs
- Difficult to add components and sensors
- Not cost effective for various types of research projects
  - Drones which meet the criteria are too expensive for many research projects

Design Choices
- Open-source drones already exist
  - Crazyflie [2]
  - pixhawk [3]
- BUT...
  - Limited weight carrying capacity
  - Limited flight time due to battery capacity
  - Microcontroller performance limitation
- We set up a development platform to allow for more sensors and devices to be added in the future
  - Camera for SLAM [4], [5]
  - GPS [6]
- So we decided to use a frame kit to build a custom drone

Architecture of Drone Flight Stack

Hardware Overview
- Raspberry Pi 3 Model B+
- Emlid Navio2 Hat for Pi
- ESC (Electronic Speed Control)
- 935KV motors
- GPS/COMPASS receiver
- 3000 mAh 3S LiPo battery
- 915 MHz Ground-to-Air Telemetry communication
- Features of Navio2:
  - Dual SPI
  - Triple-redundant power supply
  - High resolution accelerometer

Navio2 HAT Setup

Drone Operating System
- Real Time Operating System (RTOS)
  - Minimal, if any, latency in response
  - Kernel tasks can be pre-empted
  - Most not open-source
- We had the choice of using Linux [7] or Robot Operating System (ROS) [8]
  - ROS is a specialized OS for robotics
  - Due to community support and documentation, we chose Linux
- Setting up Linux for drone hardware
  - Built Linux with removal of patch to achieve nearly identical performance to a RTOS
  - Removed patch altering kernel scheduler to preempt all processes
  - Used for incoming MAVLink packets

Flight Controller - GCS

Firmware Switching
- Commercially available drones from Boeing, DII, and Skydio are capable of changing their missions mid-flight [9]
  - They are unable to completely shut down their autopilot binary and load a different one since access to their autopilot architecture is limited
- Achieving this ability would open up the field of general purpose drones to the mass consumer and industrial markets

Flight Testing Methods
- Manual Flying and Testing
  - Weather dependent
  - Battery limitation
  - Approval Process
- Simulations
  - Software in the Loop (SIL)
  - Hardware in the Loop (HIL)
- SITL simulations used to test flight code
  - ArduCopter natively compiles for SITL simulation
  - Less system resource heavy
- Microsoft AirSim [10] for HIL simulation
  - Open-source
  - System resource heavy
  - Provides environment simulation (neighborhood, city)

Performance Metric Measurements

Conclusions & Next Steps
- Conclusion:
  - Running additional workloads would decrease drone flight time, but increase response time from the autopilot software, underscoring the need for better optimizations
- Next steps:
  - Use the obtained metrics to further develop the autonomous drone
  - Conduct further research into optimizing the flight-stack and drone collaboration for deep-learning tasks (11), (12)
  - Continue building a baseline model for a general purpose drone capable of switching between firmware versions and changing missions mid-flight