Directed Research Group Report

Aryik Bhattacharya, Sujita Shah,Tom Van Deusen

May 17, 2017

1 Executive summary

There are two projects we worked on, SMACCMPilot and AOS. During the first half of the semester we worked on the SMACCMPilot project and during the other half we worked on AOS and PLEXIL plans. In the SMACCMPilot project we mostly worked with hardware setup for the copter. Afterwards we transitioned to working with AOS and began learning about SIL setup on top of ArduCopter. We used a Debian VM and VirtualBox for the entire setup and debugged various issues with setup. We further discussed all the issues we were having with the NASA team and could not get SIL to run properly on the lab machine.

2 Problem description

2.1 SMACCMPilot

The first distinct part of the project involved work done previously at Rockwell Collins, the University of Minnesota, Galois, and other industry partners on the SMACCMPilot project. SMACCMPilot is a research project to create open-source autopilot software for UAVs using high-assurance software methods that can guarantee the safety and security of the software. [1] More specifically, the SMACCM (Secure Mathematically-Assured Composition of Control Models) project was born from the idea that UAVs are relying more and more on network connectivity, and therefore are exposed to security risks similar to those faced by personal computers and other network connected devices. To combat this, SMACCM relies on the application of formal methods to prove the security and robustness of the entire UAV architecture, from the seL4 Microkernel, the only microkernel formally verified from end to end, to the flight controller [2].

Our contribution to this project was simple. We were to take a stock IRIS+ quadcopter and replicated the SMACCMCopter build created at Rockwell Collins for testing and demonstration. Given various sensors and components, we were to set up the hardware and software of the quadcopter so that it could be flown and used in demonstrations like the original quadcopter built out at Rockwell Collins.

2.2 NASA AOS

The second part of the project involved a NASA feasibility study on autonomous UAVs in shared (civilian) airspace without ground control teams. NASA has created a platform for UAVs called the Autonomy Operating System (AOS), in part based on work done for their satellite missions, and their feasibility study is largely focused on testing and building this system. Specifically, they are trying, "to determine whether a reusable software platform incorporating artificial intelligence could enable a wide variety of smart Unmanned Aerial Vehicle (UAV) apps – similar to the iPhone Operating System (iOS) for smartphone applications." [3]

Our overall objective was to create one of the aforementioned UAV apps to run on AOS. This process can be broken down into the following steps:

1. Set up an AOS development environment in the CriSys lab.
2. Review the documentation and example apps in AOS to gain an understanding of the system.
3. Review FAA documentation for inspiration on applications.

[1] http://smaccmpilot.org/
[2] http://crisys.cs.umn.edu/smaccm.shtml
[3] https://ti.arc.nasa.gov/news/AOS-ATC-test/

4. Develop and test the application.

Being that this is a bleeding edge system that is still under development at NASA, we expected to run into issues. A large part of our research, then, would be documenting the errors that we run into and reporting them to NASA.

2.3 Summary of Problem

The main goals of our project were to replicate the UAV build created by Rockwell Collins for the SMACCMPilot project and develop some sort of application for NASA's AOS. Depending on how those goals progressed, future extensions of the work could involve trying to port AOS to run on top of seL4 and SMACCMPilot, thereby improving the security of the AOS system and proving that teams outside of NASA can do extensive development for and on the AOS platform.

3 Project Summary

To begin the project, we tried to become familiar with what the SMACCM team was doing and how they modified the IRIS+ quadcopter to suit their needs. We spent time with the team at Rockwell Collins first to get an initial understanding of the modifications and then to get assistance in assembling and adding the modifications to our quadcopter. After the quadcopter had been settled, we turned towards NASA's AOS and began familiarising ourselves with it. There was an expressed interest from NASA in finding a process to incorporate AOS and running PLEXIL plans on top of the Sel4 microkernel used in SMACCM, and it became our goal to learn how to use AOS and PLEXIL in order to understand how it would be run in the SMACCM quadcopter. A Debian VM was established to house a local AOS build, and we attempted to use the Software-in-the-Loop simulator to test simple PLEXIL plans. A roadblock was hit at this point where we were not able to resolve an issue with getting SIL up and running, and we were not able to make further progress before the end of the semester.

4 Documentation

4.1 SMACCMPilot

The process of uploading the SMACCM code to our quadcopter's flight computer was a relatively smooth process because of the guides online written by the SMACCM team. In each Github repository that is relevant to preparing and upgrading the hardware, there are installation notes written by the development teams that explain the process of flashing each piece of hardware on the IRIS+ quadcopter, and building the SMACCM code base using a Vagrant VM. Personally, we ran into dependency issues because of using a newer version of Python that does not support a certain module needed to flash the 3DR radio. We found that any version of Python 3.0 would cause issues because of this dependency issue. The steps to prepare the quadcopter hardware are listed in the smaccmpilot-hardware-prep repository found on Galois' Github page.

4.2 NASA AOS

For building AOS, documentation written by the development team at NASA was given to us in order to simplify the process. It was still relatively simple to prepare a Debian VM according to their specifications, and then building AOS from source in this VM. When attempting to build and run the Software-in-the-Loop simulator, we ran into an error of a port not being open to communication. Because we have tried rebuilding the VM, AOS, and SIL on two machines multiple times, we believe this is an issue with SIL itself and have not been able to find a solution for this. Some of the issues we ran in setting up SIL include:

- PYMavlink requires the python module future, which is not automatically installed. This is easily remedied by using "pip install future pymavlink" instead of just "pip install pymavlink"
- The NASA documentation lacks information on *why* and *how* things are done, and is instead composed of prescriptive instructions on how to install the software. This makes it difficult to debug issues with installation.
- It is not entirely clear how various components of AOS talk to each other. Specifically, it is not clear what part of AOS the SIL communicates with and how these components communicate. I suspect that this communication was causing the issue with getting SIL running

- We ran into the following errors when we ran sim_vehicle.sh:
  - "timeout setting PARAM to VALUE" where PARAM is any of the mavlink parameters and VALUE is a numeric value.
  - "Failed to connect to tcp:127.0.0.1:5760 : [Errno 111] Connection refused" Interestingly enough, we ran into the same issue when we tried to install ArduPilot's SITL on a clean VM. Online searches did not reveal a solution to this issue.
  - We never reached the expected output of

    Received 412 Parameters
    Saved 412 parameters to mav.parm

    from sim_vehicle.sh.
- Note that these issues were not solved by importing the prebuilt .ova VM supplied by NASA.

5 Conclusion

Our initial goals were to duplicate the SMACCM quadcopter built by the team at Rockwell Collins, build and install AOS and SIL, and develop some novel applications for AOS. While we succeeded with our copter build, with a successful test flight completed using our copter by the Collins team, we did not succeed in building SIL and AOS. Because we could not get SIL and AOS running on our machine, we were not able to develop anything for AOS.

6 References

- http://smaccmpilot.org/
- http://crisys.cs.umn.edu/smaccm.shtml
- https://ti.arc.nasa.gov/news/AOS-ATC-test/