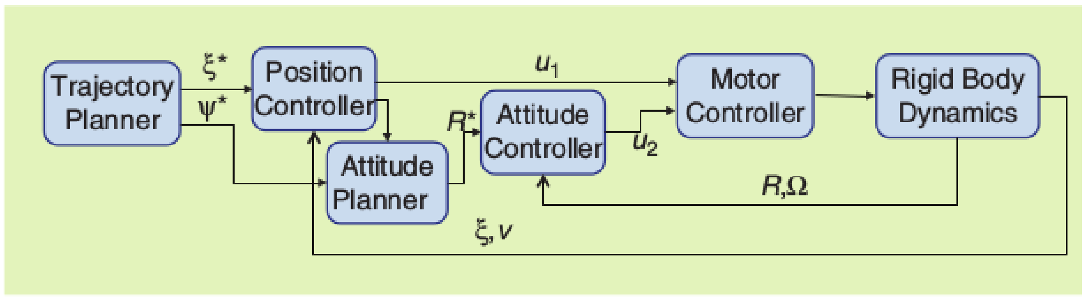
Rohan Thakker

Team C: Column Robotics

Teammates: Job Bedford, Cole Gulino and Erik Sjoberg

ILR05

Jan. 28, 2016



Mahony, Robert, Vijay Kumar, and Peter Corke. "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor." *IEEE Robotics & amp amp Automation Magazine* 19 (2012): 20-32.

Figure 1: Control Architecture

1) Individual Progress

For this sprint, Cole and I were working on the low level software interfacing of the Odroid with the PIXHAWK on the IRIS+ quadrotor. During our FVE, for the IRIS+ demo, we showed low speed communication (56700 baud per second) between the Odroid and PIXHAWK using the APM software stack. We need to set up high-speed communication (921600 baud per second) in order to run our controller at high frequencies.

After speaking to previous MRSD team that worked on quadrotors, we decided to make a mandatory non-functional requirement to have a “Kill Switch”. This is an emergency switch that can cut down power to the motor with a delay of less than 1 second. The default process to disarm the quadrotors can take about 5-10 seconds. Hence, this will help us to switch off the quadrotor in case of emergencies which are likely to occur during testing. We have successfully implemented and tested the “Kill Switch” functionality.

I also read about the control architectures used in quadrotors. Figure 1, shows the architecture of the cascaded control which is used on most quadrotors. The reference trajectory is generated for the x, y, z position and yaw of the quadrotor. The position controller calculates the reference attitude and sends it to the attitude planner. The attitude planner generates a smooth trajectory to reach that reference attitude. The attitude controller follows this trajectory by running a PID a loop. We have decided to implement the position controller on the Odroid at a low frequency (10-50 Hz) and the attitude controller on the PIXHAWK at high frequency (>200 Hz).

Next, we completed the interfacing of the optical flow sensor (PX4FLOW) to get the visual odometry updates. PX4FlOW is interfaced with the PIXHAWK using I2C and PIXHAWK is interfaced to the Odroid using UART. We had to update the firmware of the PX4FLOW to get position estimates on the Odroid as a MAVROS message (middleware to convert MAVLINK messages to ROS messages).

2) Challenges

- We faced many problems in increasing the baudrate of the PIXHAWK from 57600 to 921600 bauds per second on the TELEM1 port. This was mainly due to lack of proper documentation of the APM software stack. After browsing through many blogs and forums we realized that the APM stack was not designed for researchers. PX4 is another popular software stack that is used by many research labs which work on controls. Further, the documentation of the PX4 is much better than PIXHAWK.

  Hence, Erik started testing the PX4 stack on our backup IRIS+. He was able to get all the settings done within a few hours and test it on the backup IRIS+. Once we validated that the PX4 software stack is stable, we shifted both the quadrotors to it. Further, the latest version of the PX4 stack had an implementation of the "Kill Switch" and this feature was well documented and tested. Thus, we did not have to waste a long time to implement this feature. We were also able to configure a switch on the RC remote to control the switching between manual and auto mode.

- We also faced many problems while interfacing UART on the Odroid due to lack of proper documentation of the USB-UART pins. Eventually, we solved the problem by shifting to the UART pins located on the GPIO Connector-10 because documentation was available for these pins. We tested the connection by connecting the Tx and Rx pins of the UART on Odroid. This was a nice trick that helped us eliminate the PIXHAWK (since we did not know whether the problem was from the PIXHAWK or Odroid).

- Before the FVE, we had ordered a level shifter that was recommended by the manufacturer for interfacing the Odroid and the PIXHAWK. Since the Odroid runs on 1.8V and PIXHAWK runs on 3.3V. However, we found that the level shifter was not working. Hence, we used a level shifter from the MRSD inventory. For this, we had to make our own connectors.

- There was some bug in the latest version of QGroundControl i.e. a GUI used to update the firmware of the PX4FLOW and the PIXHAWK. For some reason, we were not able get the PX4FLOW option in firmware update. Hence, we reverted to an earlier version of the QGroundControl, using which we were successfully able to update the firmware.

3) Teamwork

Our team got a lot of work done this sprint. Erik set up the backup IRIS+ and helped in testing the PX4 stack. Further, Cole and I have complete all the low level interfacing work. We have also shifted to the lasted PX4 stack which was compiled from source. Hence, we have the ability to make changes to the software if required. Throughout the two weeks, Erik, Cole and I were working closely to get things done on the IRIS+.

Further, Job was continued to work on ArDrone which is our risk mitigation strategy, in case we are not able to get stable flight on the IRIS+ quadrotor. He worked on pose estimation using the APRIL Tag library. This was implemented using a separate library that is not specific to the ArDrone. Hence, we will be able to use this on the IRIS+.

For the FVE, we used the TUM_ARDRONE library on top of the ARDRONE_AUTONOMY drivers. The motions were generated by writing a script in the TUM_ARDRONE GUI. This sprint, Job wrote a custom ROS node to directly send ROS messages. This will help us in implementing our own position controller which will be required for landing.

4) Plan

We intend to implement autonomous landing by the end of February on the IRIS+. Our goal for next sprint is to implement and extensively test autonomous hover on the IRIS+. Hence Cole and I will be working on getting this done. Further, Job will continue development on the ArDrone to implement autonomous landing. We decided to test this by drawing four circles representing the funnels on a chart paper and see if the ArDrone can land in those four circles. During the initial phase of this semester, we wanted to work on the things that we are not confident about implementing. Hence, we decided that Erik should work on implementing global position estimation using a SLAM algorithm on the Odroid. This will be required if the position estimation given by the PX4Flow has a lot of drift.

5) References

[1] https://pixhawk.org/modules/px4flow

[2] http://www.hardkernel.com/main/products/prdt_info.php