Real-Time Linux Quadcopter - Technical Takeaways

Charlie Sands - September, 2022 - Northville, Michigan

Overview

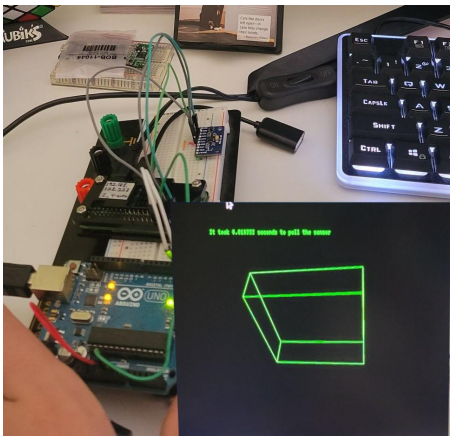
My quadcopter with the protective flight controller cover removed

While studying the Linux Kernel in order to become a contributor I became interested in the kernel process scheduler. More specifically, I wanted to understand how Linux was being used in industrial environments for automation and systems control. I decided to modify my old DJI 450 airframe quadcopter in order to use a custom flight controller based on the PREEMPT-RT framework running under Linux. Running under an instance of BuildRoot, I wrote real-time code that runs under the preemptive scheduler to manage the real-time systems on quadcopter, such as angle detection, motor control and auto leveling. The standard Linux "Completely Fair Schedule" manages the network stack, vehicle movement command scheduling and other non-time critical tasks. I also experimented with writing my own driver for motor control, however, the final product did not implement my driver for stability reasons. The onboard Linux computer used in the quadcopter is a Raspberry Pi Zero W. All hardware management is done over I^2C.

Algorithms and Control Schemes

Running a single tethered rotor on my "PID test" jig

The quadcopter uses a combination of algorithms to control its heading, attitude and altitude. I implemented many proportional integral derivative (PID) control loops in order to manage the onboard systems. There are seven PID controllers used on the quadcopter. The first three controllers are for pitch, roll and yaw control while the following four control motor speed for each individual rotor. These controllers are not, strictly speaking, necessary, however, they should (in theory) reduce strain on the motors by not abruptly jumping between motor speeds. The system is optimized such that if the requested change in rotor speed is small enough, the controller is deactivated to save valuable processing time. In order to deduce the pitch, roll and yaw location of the quadcopter I implemented the Madgwick inertial measurement unit algorithm in my code. This algorithm fuses the data from the quadcopter's accelerometer, gyroscope and compass to determine an appropriate heading and attitude of the aircraft. This data can then be used to correct to calculate the error between the current and target aircraft positions. The mathematical concepts that allow this to work are far beyond me at this point, but my implementation, based on those of others, worked well.



Testing the IMU algorithm with a 3D box corresponding to the position of the control board prototype

Safe and Reliable Systems

In order to ensure the safety of people around the quadcopter I incorporated a network ping system that ensures that if a WiFi connection is lost with the computer controlling it, nothing bad happens. In addition to this I had to ensure that all best-practices were followed. Steps such as clearing all of the PWM controller's pulse width control registers when disabling the quadcopter are important to ensure that the rotors do not erratically spin up when the aircraft is re-enabled.

Approximate Cost	$300
Approximate time spent	3 months
Skills developed	- Closed Loop Control Algorithms - Embedded Linux Development - Avionic Systems - Safe and Reliable System Design