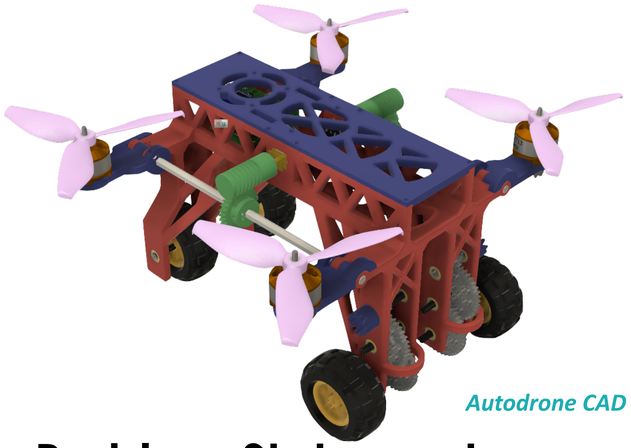
Autodrone – Multi Domain Vehicle

Midn 1/C Wolfe

ROBOTICS AND CONTROL ENGINEERING
UNITED STATES NAVAL ACADEMY

Motivation

Current autonomous vehicles are either airborne or terrestrial, limiting their ability to efficiently navigate in complex environments. The goal of this project is to develop a proof-of-concept drone that can transition between ground and flight modes using a single set of motors for propulsion.



Autodrone CAD Model

Problem Statement

How can we create a platform that can move large distances quickly while also being maneuverable in small spaces. In addition, how can we make a vehicle that is quieter than a conventional aerial vehicle. Lastly, how do we make this platform robust so it can perform in a variety of climates and be able to carry the weight of sensors the user might want.

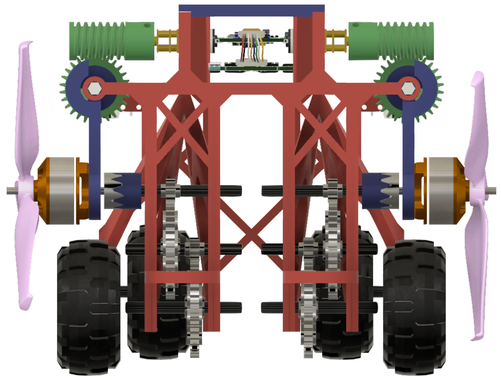
Related Work

The Sharper Image Fly is a drone that can fly and drive. It has separate motors for the driving functionality. This product is marketed as a toy.

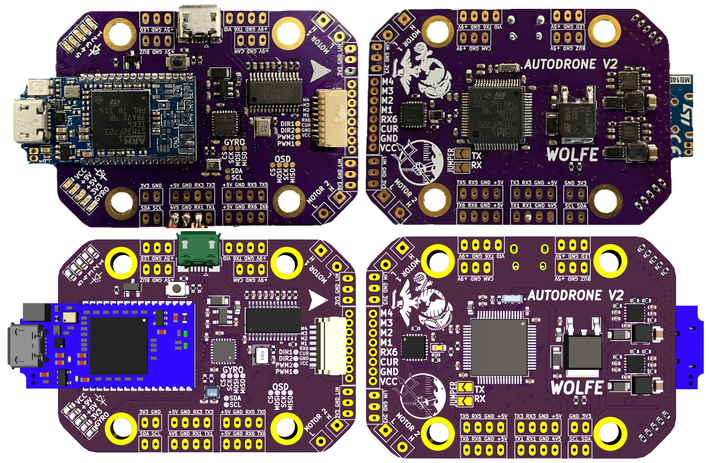
Sharper Image Fly

Methods

Mechanical Design: The frame of Autodrone was designed in Fusion 360, a 3D CAD tool. This tool allowed me to design components and then define interactions between them. I was then able to simulate things like the transition between modes as well as the gearbox connecting the motor to the wheels. This allowed me to better conceptualize how my design would work in practice so I could develop a more robust model before sending it to the 3D printer.



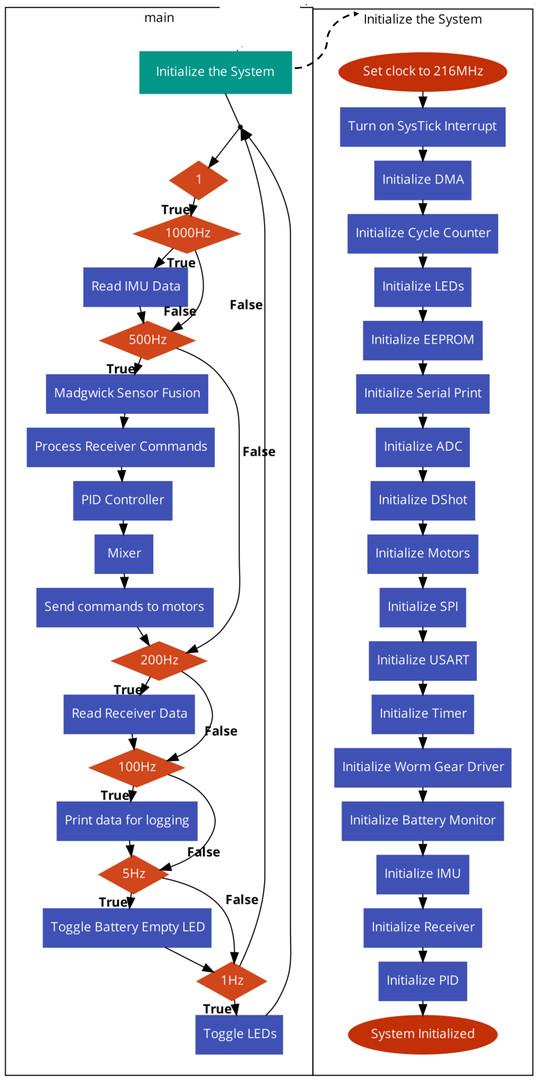
Electrical Design: The printed circuit board(PCB) for Autodrone was designed in KiCad. It has six layers, using an STM32F722 microcontroller as the central processor. An MPU-6000 6-axis inertial measurement unit(IMU) provides accelerometer and gyroscope data. It also embeds an ST-Link for easy debugging over the serial wire debug(SWD) interface. Other features include, a barometer, an on-screen display(OSD) chip, +9V, +5V, +3V3 outputs, and a brushed motor driver.



Hand-built PCB and 3D Model

Methods Cont.

Software Design: Autodrone is coded entirely in embedded C, meaning it interacts directly with the hardware of the microcontroller. No online libraries or drivers were used in the writing of the code. I chose this path mainly to get a true understanding of how a flight controller works at a low level and learn more about embedded systems. This approach tends to also be much more efficient, saving valuable hardware resources and consuming less power.



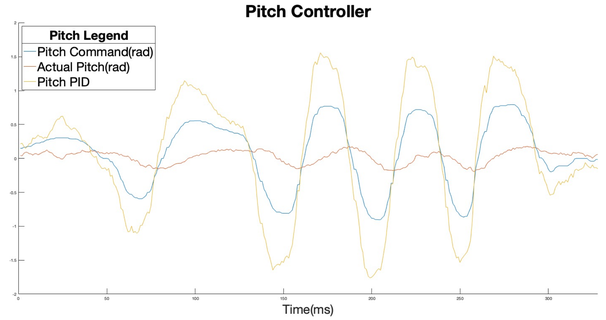
Autodrone Code Flow Diagram

Results

Autodrone is capable of terrestrial navigation as a rover, converting into a quadcopter, flying, and then converting back to a rover.

Scan the QR code to the right to be taken to Autodrone's page. There, you will find my code documented by Doxygen as well as a PDF of my PCB schematic.

Below is data taken from an onboard logger during a test flight. This plot is recording only data in the pitch axis. You can track the actual vs desired pitch states as well as the PID gain calculated to minimize error.



As you can see, the actual pitch does not follow the desired pitch very well. This is an area I hope to improve.

Conclusion

I believe that I met the goals that I set out to achieve during my initial idea formulation phase. Autodrone is a launching point for future projects. I see future contributors adding capabilities like, enhanced sensor integration(IR, RADAR, etc.), an advanced PID controller for improved flight performance, integrating a GPS and barometer for altitude and position hold, or even an ability to slightly rotate the arms in flight for thrust vectoring. The opportunities are limitless for Autodrone.

Acknowledgement

I would like to thank all the techs at TSD, Joe Bradshaw, Dan Rhodes, John Seargeant, and Patrick McCorkell, for helping me with the everything from the mechanical design of my frame to sourcing parts and tools. All the professors I consulted with in WRC and the ECE departments: Professors Bishop, Broussard, Dawkins, DeVries, Elsberry, Feemster, and Ngo. Outside of USNA, I would like to thank Pete Smits, the developer of AM32 Multirotor ESC Firmware for spending countless hours guiding me through the many problems that come with embedded programming.