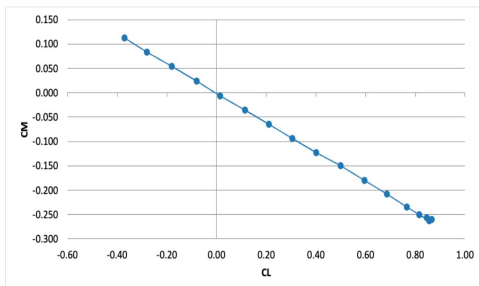
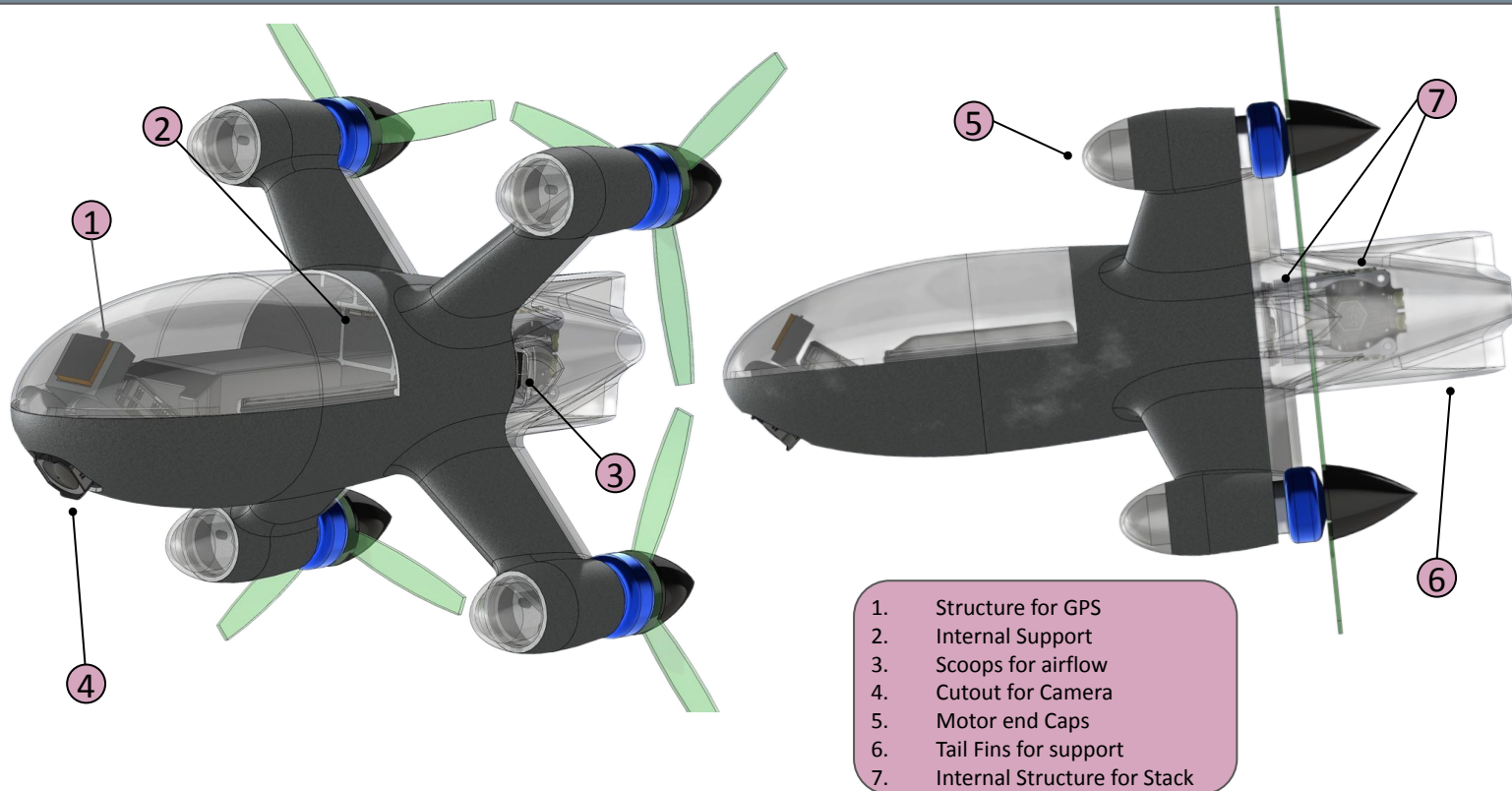
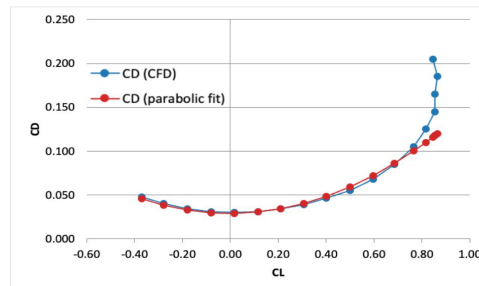




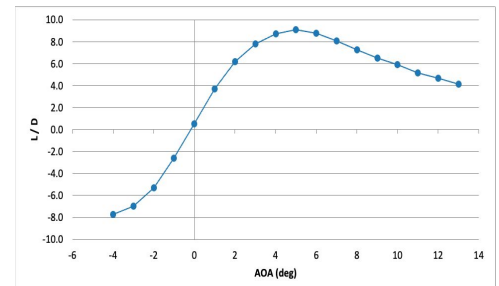
300+km/h Quadcopter Speed Drone (Design, CFD and Iterations)



CM vs CL - Static Margin = 0.30

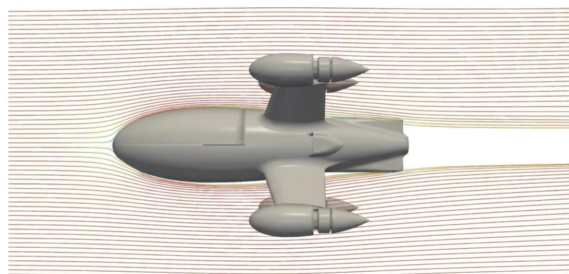


Drag Polar With Parabolic Fit

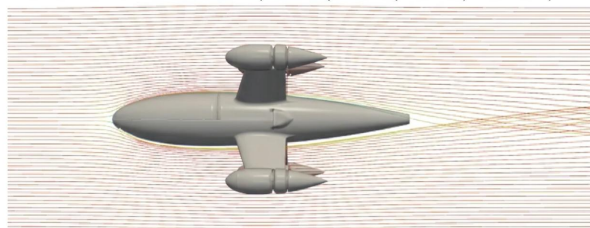


L/D vs AOA - Peak 9.1 at 5 deg

Side | Cd Total | Cd Pressure | Cd Viscous | Length |
Original Version (Current) Cd = | 0.127 | 0.091 | 0.036 | 355mm |



Side | Cd Total | Cd Pressure | Cd Viscous | Length |
Low Profile Version 2 - Cd = | 0.144 | 0.102 | 0.041 | 440mm |



Surface Pressure Contours

Streamlines at 0° AOA - longer body separates earlier at the tail junction.

What?

I designed a quadcopter drone optimised for speed and ease of production using stock 5" parts. I drone is 3d printed out of regular PLA and includes GPS for future autonomy implementation

How?

Built the airframe in SolidWorks around the electronics and the camera stack. Solved the external flow in OpenFOAM across angles of attack from -4° to 13° and speeds from 10 to 100 m/s, inspecting pressure and separation in ParaView. Reduced the output into a drag polar, stability slope, and efficiency curve, then compared four fuselage shapes and checked power demand against what the motors deliver.

Results

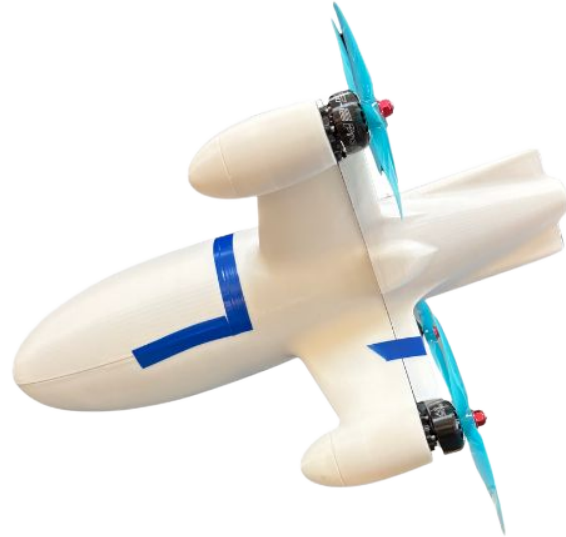
CAD and print complete at 880 g
Drag polar: $CD_0 = 0.029$, $k = 0.122$
Pitch-stable passively - moment slope -0.30
Peak L/D of 9.1 at 5°
Departure at 12° , $CL_{max} 0.865$ - sets controller limits
Camera cutout costs Cd $0.109 \rightarrow 0.172$
1.0 kW needed at 300 km/h against 3.4 kW installed
Ceiling set by propeller pitch, not drag



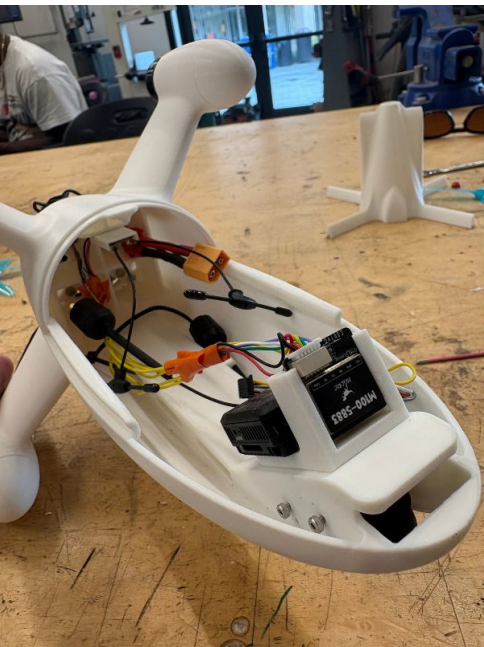
300+km/h Quadcopter Speed Drone (Build and Flight)



Maiden Flight - achieved 172 km/h going 48 percent throttle



Final assembly pre test flight



Electronics Integration into Final Print



All previous iterations

What?

I built and successfully flew the drone for the first time

Will do a max thrust test when new propellers arrive

How?

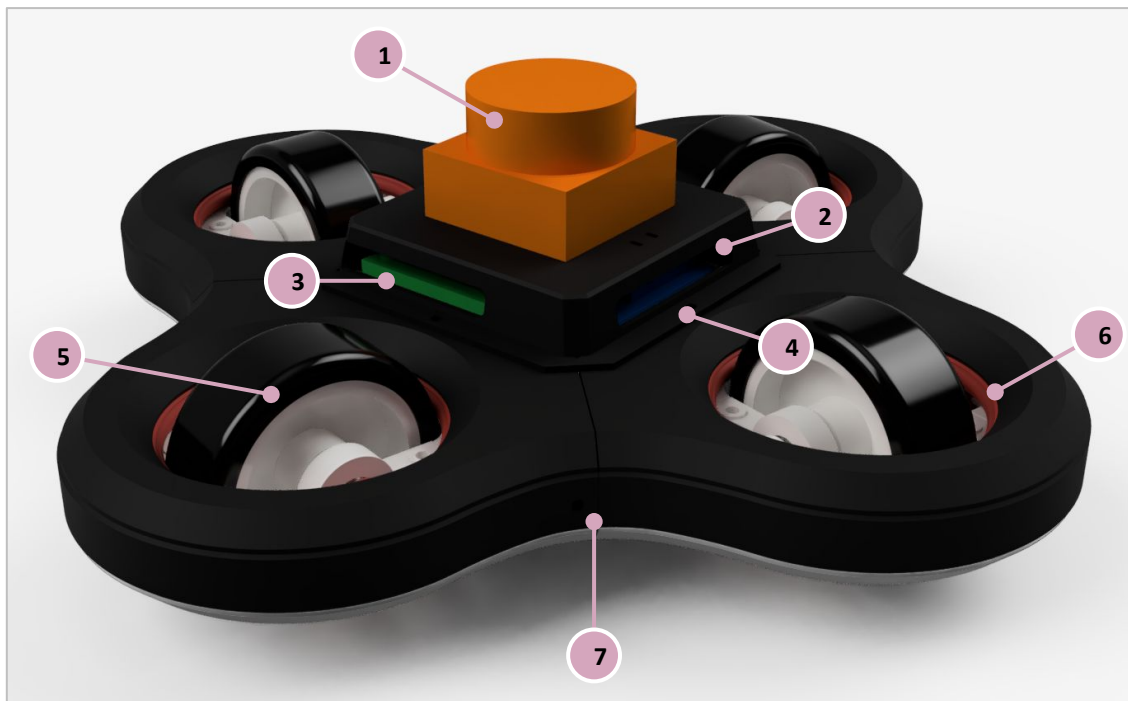
Assembled using m3 screws and nuts
- All electronic soldering was done by myself
- Used ArduPilot for the PID tuning of the drone

Results?

- GPS top speed of 172 km/h during first flight
- Around 3400 W of max Power
- Included GPS means capable of autonomous flights
- Flew 312km/h in third flight



4-Wheel Differential Swerve Robot (Design and CAD)

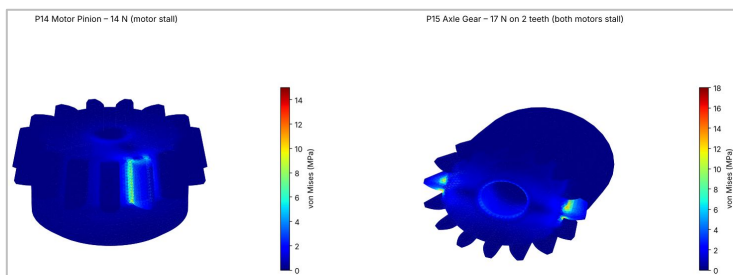


Key

- 1 RPLIDAR C1 mount
- 2 Centre cap (35 mm tall)
- 3 Raspberry Pi 5 bay
- 4 ESP32 + motor drivers
- 5 Differential swerve module (x4)
- 6 Ring gear and hold-downs
- 7 Four-piece printed chassis



One module: two N20 motors both steer and drive the wheel



Gear FEA at motor stall: peak 12 MPa (PLA about 50 MPa)



Exploded view of the printed layers

What?

I am designing a four-wheel robot where every wheel sits in a differential swerve module: two N20 motors per module both steer and drive the wheel, so the robot can move in any direction while rotating.

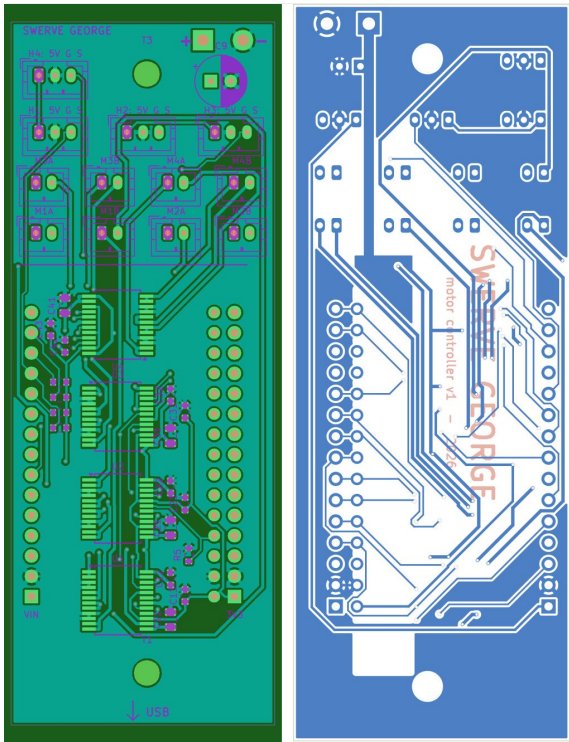
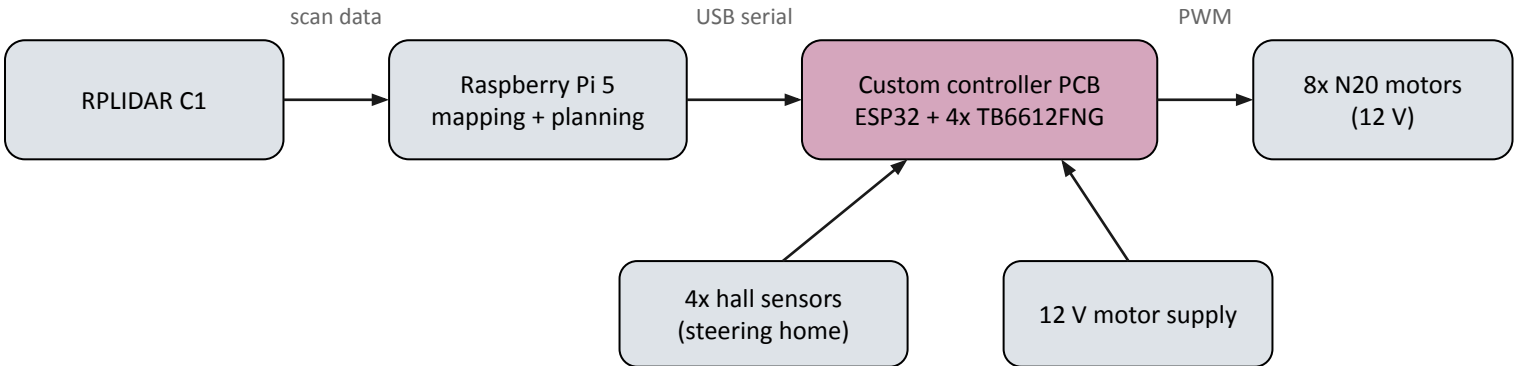
How?

Modelled the full assembly in Fusion 360 and 3D printed every structural part in PLA. Ran FEA on the gears and bearing brackets at motor stall load, then iterated the pinion bores, ring hold-downs and centre cap through several print-and-fit cycles.

Results

- 4 modules, 8 N20 motors, all parts printed
- Unlimited steering: no wires turn with the wheel
- Gear FEA: peak 12 MPa vs about 50 MPa for PLA
- Pi 5, controller and LiDAR fit under a 35 mm cap
- One hall sensor per module for steering homing

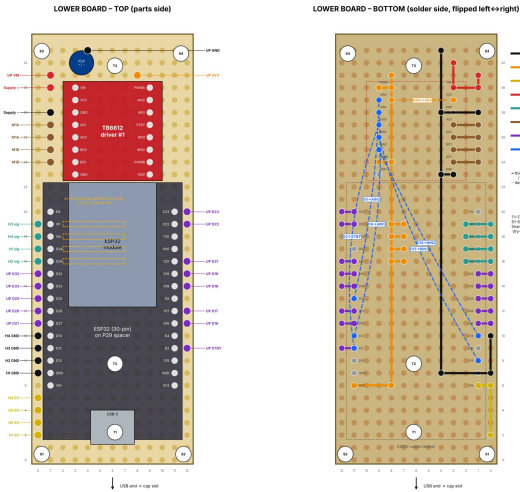
4-Wheel Differential Swerve Robot (Electronics and Custom PCB)



Controller PCB, front and back



Centre bay: Raspberry Pi 5 (green) beside the ESP32 controller (blue), under a 35 mm cap



First plan: two hand-wired perfboards with dozens of solder links. One custom PCB replaced both.

Board	88 x 33 mm, 2-layer
Drivers	4x TB6612FNG, 8 motors
Sockets	8 motor + 4 hall, JST-PH
Checks	0 DRC errors, 0 open nets

What?

Electronics for the robot: a Raspberry Pi 5 handles LiDAR mapping and path planning, while an ESP32 runs the eight motors and four hall sensors in real time over USB.

How?

Wrote the ESP32 firmware (20 kHz PWM, hall-sensor homing, calibration saved to flash). Designed a 2-layer controller PCB in KiCad, autorouted it with Freerouting, checked every net against the firmware pin map and sent it to PCBWay for assembly.

Results

- Test firmware uploaded and running on the ESP32
- One 33 x 88 mm PCB replaces two hand-wired boards
- 0 design-rule errors, 0 unconnected nets
- 5 assembled boards in PCBWay review
- Next: module test, closed-loop control, LiDAR mapping