Project Pixel Orbital

Mission

Project Pixel Orbital is a student-led team at Stanford Online School dedicated to launching a 3U CubeSat ("PixelSat I") to low-earth orbit NET November 2026.

The mission objectives include the following:

- evaluate COTS components and a partially additively manufactured (3D-printed) polymer frame to push down the cost of satellite productions
- capture and downlink one clear picture of a figure of a Pixel (OHS school mascot) from orbit with Earth in the background
- operate and relay data from a student-created protein crystallization experiment to reseearch drug manufacturing in space
- show that a team led by high schoolers, with no adult supervision and operating on a mere $10,000 of hardware budget, can successfully execute the mission as described above

Launch & orbit

The current goal is to launch the satellite on Rocket Lab Neutron rideshare in November 2026, targeting sun-synchronous orbit and a natural decay.

Resources

- public website for outreach and progress updates at https://www.projectpixelorbital.com/, run by Daphne
- GoFundMe to cover hardware costs at https://www.gofundme.com/f/support-project-pixel-orbital-student-team run by Shiv
- Canvas at https://spcs.instructure.com/courses/9094/pages/project-pixel-orbital
- Project overview at https://docs.google.com/document/d/1YouaE7v0XuAZrTZKKp6WArcYLAC4PqZ7/edit

Team

The team operates across two platforms: a Discord server and Pronto group chat (Pronto is the official school messaging platform). Meetings are every Friday at 11am on Pronto.

- **Weston Yoo** (he/him): overall team lead
- **Daphne Foroughi** (she/her): outreach team lead
- Francesca Davies (she/her): outreach team member
- **Victoria McLeod** (she/her): structural team lead
- Grace Zheng (she/her): structural team member
- Issac Goldberg (he/him): outreach team member
- Marianne Tzeng (she/her): microbiology team member, software team member
- Nathan Muruganantham (he/him): electrical team member, ADCS team member
- **Satya Kokanda** (he/him): microbiology team lead
- Shiv Sareen (he/him): outreach team member
- Shravya Jain (she/her): outreach team member
- Siyona Agarwal (she/her): software team member
- Ashwin Naren (he/him): software BDFL
- **Vinayak Vikram** (he/him, often referred to as "yappy"): electrical team lead, ADCS team lead, software team colead, comms team member
- **Aadish Verma** (he/him): comms team lead, software team colead
- Greyson Wyler (he/him): electrical team member
- Jesse Guo (he/him): software team member

- Charvi Atreya (she/her): ADCS team member

Structural

The primary requirement for the structural team, other than the 3U form factor, is the $10,000 budget, much of which is used for electrical components and other miscellaneous items. Thus, one of the key goals of the project is to demonstrate the performance of 3D-printed polymer structures in orbit as alternatives to metal frames.

The frame is 100x100x300 millimeters and planned to be 3D printed using carbon-fiber-filled polyphenelyne sulfide (PPS) plastic with 90-100% cubic infill. Rails must be square, with a 0.7mm chamfer on the outer corner. The current structure features 5 horizontal rings between the rails, forming cross-braces for attach points for the panels. The battery will be located in the -x-y corner of the frame, flush with the rail. The Pixel will be in a holder at the top of the frame. An open section in the frame allows the onboard camera to view the Pixel. The target mass is below 6 kg.

Software

> **Note**
>
> ESP32 refers to the classic dual-core ESP32, not any of the newer variants. We are using the classic ESP32 for fault resistance reasons.

> **Note**
>
> The PI and camera are under discussion, we have not standardized on whether and how they will be used.

Software is a “meta-team” that includes collaboration between several separate teams to write, verify, and integrate 1) the flight software for the satellite and 2) the software that operates on ground to link to the satellite.

The platform planned involves a dual design to account for the wide range of needs of the system, consisting of an ESP32 and Raspberry Pi Zero 2 W. In this setup the ESP32 serves as the main flight computer, handling all critical functions such as ADCS, comms, and power management. The Raspberry Pi serves as a secondary computer that handles the camera payload and the microbiology experiment (if it happens), which are less critical to the satellite’s operation and can be rebooted by the ESP32 in the event of a crash.

The ULP in the ESP32 will be used to watchdog the Raspberry Pi, and the ESP32 itself will be watchdogged by a hardware watchdog chip that can reset it in the event of a crash.

Alternatively we use a single ESP32 to handle the entire satellite, due to the PI seeming unneeded if we only use it for camera data. A camera that can output jpeg encoded images directly to the ESP32 will be used, and the microbiology experiment will be gutted entirely. The image will be stored on a partition on the ESP32 and chunked for downlinking.

Code on the ESP is primarily no_std Rust, if a PI is used the ULP for PI watchdogging will be written in Xtensa ASM or C.

If used, ESP/PI comms will likely happen over UART and the PI comms handling will likely be shared with the ESP (which means it will be written in rust).

The software team itself is more of a “meta-team” consisting of team leads and members across a wide range of other teams working together to write the flight software for the satellite. More

software details will thus be given in their respective sections for the ADCS, comms, and electrical teams.

Software is currently using a monorepo for ESP code at https://github.com/pixelsat/pixelsat. This has code for both comms and ADCS.

Comms
We are using LoRa over UHF with an SX1262 transciever from Semtech. It is unlikely that we will ever use GMSK due to the dopler shift. Likewise S-BAND is not being considered due to paperwork and infeasibility concerns.

Camera downlink will have to be be chunked somehow.

For downlink we will likely use tinygs.

ADCS (attitude determination and control system)
After deployment, the ADCS will need to kick in nearly instantly to detumble the satellite and execute several basic maneuvers to demonstrate full functionality before establishing comms contact.

Attitude Determination System
The ADS will be an always-on system, constantly running on CPU1 of the ESP32.

The proposed sensors for the ADS are outlined below:
- Combined IMU (**Gyroscope**+accelerometer) + Magnetometer (Adafruit Precision 9 DOF IMU (ICM-20948))
- Sun sensor (Solar panels as coarse sun sensors, as described in the electrical section)

The ADS constantly computes accurate attitude data by fusing the gyroscopic data with one or more of the absolute attitude-determination systems using an EKF. Clock cycles for the data fetching will be strictly controlled, and adequate compute cycles will be allocated for the EKF.

It is possible that gyro data can be fused with accel data for even more accuracy.

Attitude Control System
The satellite will use a three-axis magnetorquer system for detumbling and coarse 3-axis attitude control.

The magnetorquers will be air-core enamel-wound, with a build aiming to mimic the footprint of a flat-coil magnetorquer for a fraction of the cost. Magnetorquers will be home-wound by ADCS team members in the Bay Area.

The total power allocated to the magnetorquers has a 3W ceiling (combined), so it is prudent to throttle each magnetorquer at 1W. Wire length/number of turns is left TBD as of right now, as the bus voltage to the torquers is unknown.

Flat-coil/air-core magnetorquers operate by creating a magnetic dipole moment m orthogonal to the plane of the magnetorquer. The magnitude is given by $m = N \cdot I \cdot A$, where N is the number of turns, I is the current, and A is the average area of a single turn of wire (note that this is the average of the innermost and outermost turns). Torque is simply calculated as $\boldsymbol{\tau} = \boldsymbol{m} \times \boldsymbol{B}$.

As of right now, each torquer will be constructed by winding copper wire around a central rectangular bobbin with guardrails to make sure the wire does not slip off the top/bottom. Note that though this is technically an air-core magnetorquer, the design will be as such that they fit exactly as a flat-coil magnetorquer would and saves us significant cost in PCB production (can range to over $3000 for this specialized an application). Note that our bobbins can **not** be constructed with PLA or

PETG; both will soften in our use case. We would have to use polycarbonate or PEI. It is also probably prudent to create a jig to wind the torquers, as doing it by hand could lead to many inaccuracies.

Simple open-loop PWM would not be acceptable in this context since resistance increases as the wire heats up as well as the fact that the battery output fluctuates, so we would need to implement closed-loop PWM with a current-sensing resistor or a digital monitor of some sort. Obviously, the ESP cannot send 11.1V over a bus, so we use an H-bridge (a combination of MOSFETs) to allow the ESP to flip the 11.1V from the battery across the coil to generate our torque.

In the final deployment, we will probably have a routine on the ESP to apply a 3-axis torque; it will use magnetometer data to estimate $\boldsymbol{B}$ at the point and generate the necessary magnetic moments through all the torquers, varying over time by again using an EKF. There is a chance we opt for a GPS-driven WMM; however this is *extremely* unlikely since this will significantly throttle the ESP and this work cannot be reasonably offloaded to the main computer.

Simulation pipeline

Note: the simulation pipeline is mostly for ADCS; it serves as an interface to test routines before the actual launch. The pipeline uses Zenoh for messaging between simulation nodes and is written for Basilisk, an astrodynamics simulator that uses Python to interface with a high-performance C++ backend. Simulator nodes publish satellite updates and accept control commands through the Zenoh-based sim network.

Outgoing state updates mostly contain satellite attitude/orbit data (for testing suites, etc) as well as raw virtual sensor data that will be processed in the routines. Incoming commands at the moment are `applyTorque` and `applyMagneticDipole`.

Electrical

The electrical subsystem mostly serves the role of power management across the satellite. Master power is delivered by a lithium-ion battery pack configured as 4P3S (4 parallel, 3 series) 21700 form-factor cells (12 total cells). Each 4P cell group is Kapton-tape-wrapped for stability, protection, and insulation. The satellite-wide voltage bus will be at 5V.

Power is distributed accordingly across the following systems by a power controller (custom-designed from a relay network, probably), ranked by priority in the event of low power:

1. Onboard microcontroller
2. Attitude *determination* system
3. Communications radio
4. Attitude *control* system
5. Onboard computer
6. Camera payload

Heatsinks+passive battery warmers via CPU heat (negligible with our current hardware) redirection are being considered, alongside aluminized thermal blankets as insulation. Battery performance may fluctuate or drop due to temperature changes in orbit.

Radiation tolerance is not yet fully evaluated. However according to a goated European unc, the ESP32 should survive for somewhere from a week to a month.