Autonomous Self-Balanced Bicycle

https://github.com/LuckyYoon/autonomous-self-balancing-vehicle Justin Yoon

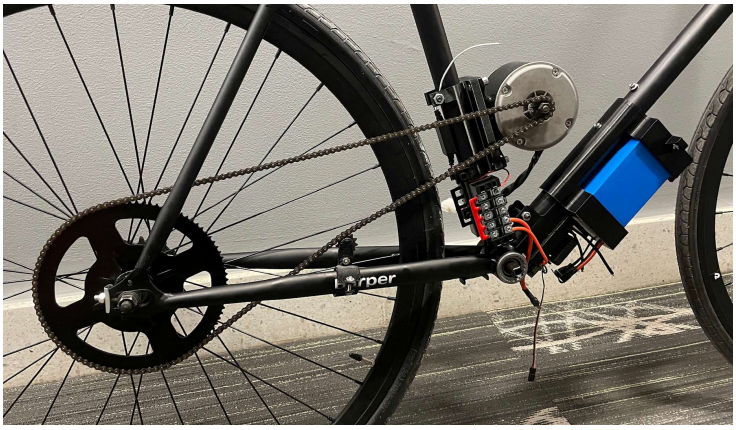
Motivation

This project started from a long standing interest I've had since high school in building an autonomous bicycle. An autonomous bicycle requires nonlinear control and autonomous navigation, which makes it a compelling topic for research and experimentation (and it makes for a fun project). Several prior works shaped the direction of this build. The Xuan Bike by Peng Zhihui demonstrated that reaction wheel based balancing was viable on a full scale bicycle and provided a practical reference for the stabilization approach used here. Google's autonomous bicycle April Fools video was an early reference for where I got this idea. The RAI Ultra Mobility Vehicle demonstrated how this idea could be expanded upon, and reinforced the approaches like steering based balance and control moment gyros. The broader goal was to produce a platform that is genuinely accessible. The bare configuration comes in under $1,000, utilizing 3D-printed components and off-the-shelf hardware.

Hardware Choice

The Bill of Materials and justification for the full configuration can be found at https://github.com/LuckyYoon/autonomous-self-balancing-vehicle/blob/main/docs/bill_of_materials_full.md

Build

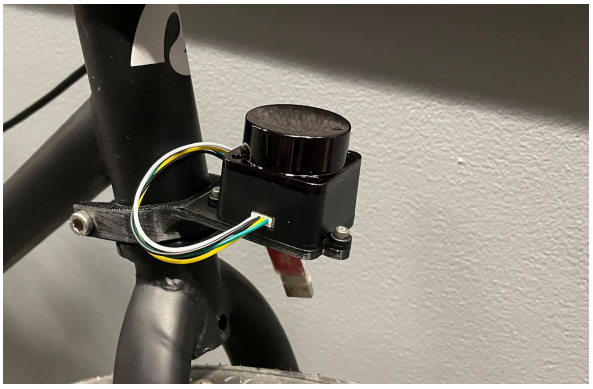


The build started by stripping the bicycle down to its frame. The original drivetrain and brakes were removed. The drive motor handles braking through regenerative deceleration, so a mechanical brake would add weight without function. The brushed DC drive motor was installed with a 25H chain drive to the rear wheel using a 90T/11T sprocket reduction. The drive ESC and main 24V battery were mounted as low on the frame as possible to keep the center of mass low and reduce the angular momentum demands on the reaction wheel during disturbances. A terminal block was also installed at this stage to serve as the power distribution point for all electronics.

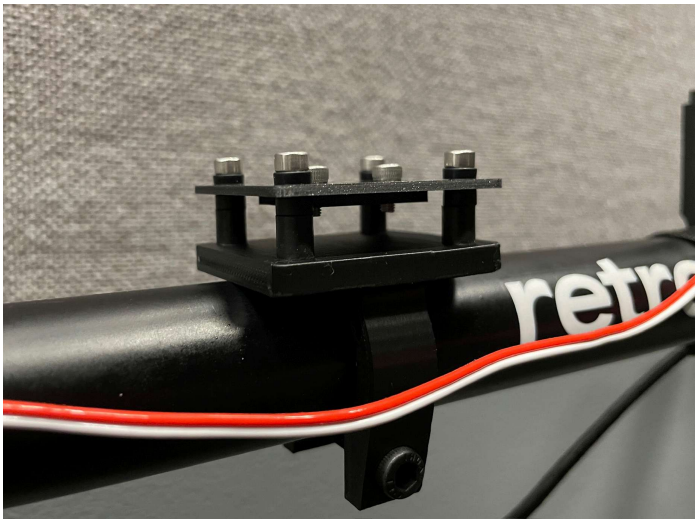
With the handlebars removed, the headtube nut was accessible for mounting the 3D-printed steering gear. The 160KG servo is mounted to the frame and meshes with this gear to actuate the fork. The headtube sits at a 16° angle from vertical. The servo mount was designed with a matching 16° bevel so the gear faces are parallel and the mesh is clean.

The electronics mount holds the Jetson Orin Nano, the 24V-to-12V buck converter, the STM32F411, and the ODESC V4.2 motor controller. It is double-sided to reduce the overall footprint. The mount went through one significant design revision. The initial orientation was horizontal (flat, parallel to the ground). This was convenient during assembly but problematic in practice: when the bike fell on its side during testing, the horizontal mount which stuck out, took the impact and broke.

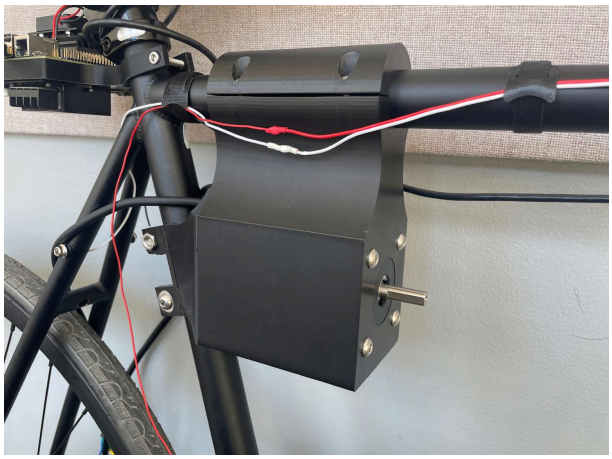
The mount was redesigned to sit vertically. In this orientation, a fall strikes the edge of the seat rather than the mount, reducing the chance of electronics damage. The vertical layout also improved accessibility to connectors and ports during development.

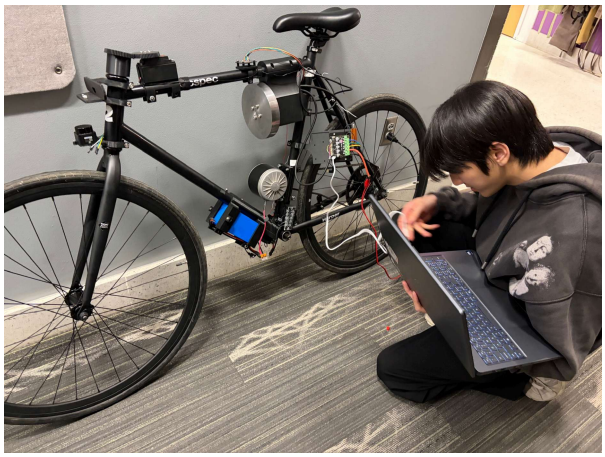


The ZED 2i stereo camera is mounted at the front of the bike for a forward facing visual odometry. The 2D LiDAR sits below the camera for obstacle detection. Both use the same 3D-printed tube clamp mounting system as the rest of the platform.



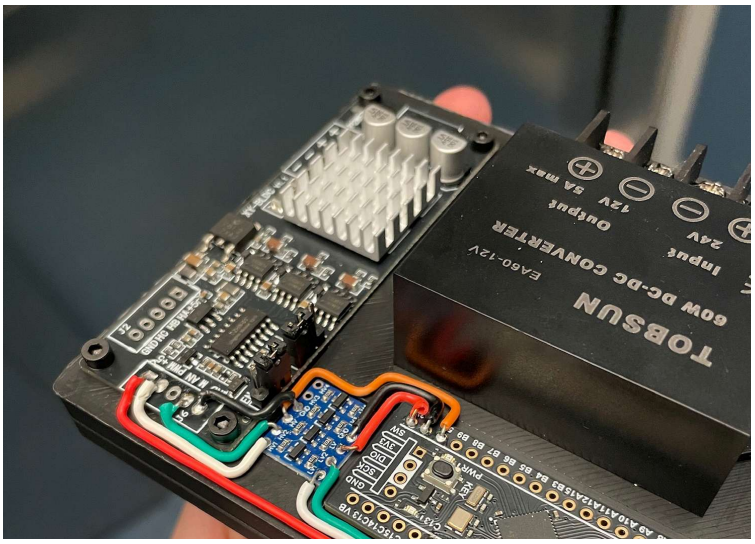
The IMU required a dedicated mount because the quality of the roll angle measurement directly affects balance performance. High frequency vibration from the motors and drivetrain can corrupt the IMU signal and degrade the performance of the reaction wheel. The mount uses rubber M3 anti-vibration standoffs, which mechanically decouples the IMU from frame vibration before it reaches the sensor. The mount is double layered: the IMU attaches to an inner plate, which connects to the frame through the rubber standoffs on an outer plate.



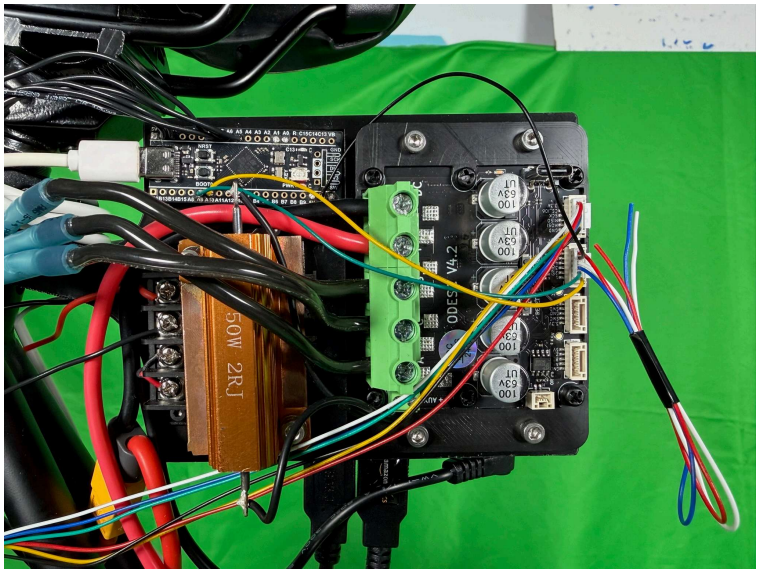


The original reaction wheel motor was a 24V 200W NEMA24 BLDC. While the power rating seemed adequate, it was not able to handle the rapid high frequency direction changes. The motor was paired with a solid disk of steel 6 inches in diameter and 1 inch in thickness. This flywheel turned out to be too heavy for the motor.

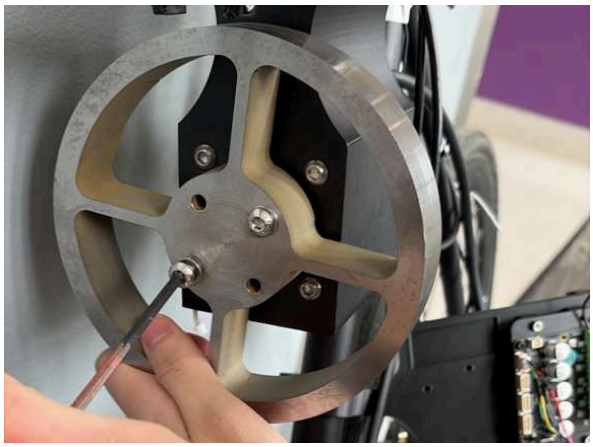
The replacement was the ODrive M8325s 100KV brushless outrunner. The lower KV rating translates to higher torque at lower RPM. The larger rotor diameter of an outrunner geometry also increases rotational inertia and torque per amp relative to an inrunner of comparable size. Peak torque is 6.6 Nm, the motor handles the high torque that fast reversals demand.



The initial motor controller was an XY-BLDC 200W driver. The problem was the delay on direction reversal commands, it could not execute an instant direction change. For a balance loop running at high frequency, any perceptible lag between a torque command and motor response leads to destabilization. The XY-BLDC was replaced with the ODESC V4.2, which meant this beautiful wiring that I may or may not have spent 2 hours on, had to go.



The ODESC V4.2 is built on the ODrive platform and supports Field-Oriented Control (FOC). FOC controls the magnetic flux and torque vectors in the motor independently, which enables precise low-latency torque commands rather than speed or voltage commands. This is the correct control mode for a reaction wheel application since we care about the acceleration of the wheel, not the direction in which it's spinning. To support the aggressive direction changes during operation required for balance, max_regen_current and max_negative_current were both set to approximately 80A, making the controller allow the high current spikes without any errors.



The wheel was milled to remove material from the center, concentrating the remaining mass at the rim. This maximizes the moment of inertia ($I = mr^2$) for a given total wheel mass, since inertia scales with the square of the radius from the rotation axis. The result is a wheel that stores more angular momentum compared to its total mass.

Active Reaction Wheel

The STM32F411CEU6 handles the computation for the reaction wheel. This includes reading from the IMU through SPI and commanding motor torque to the ODESC through UART.

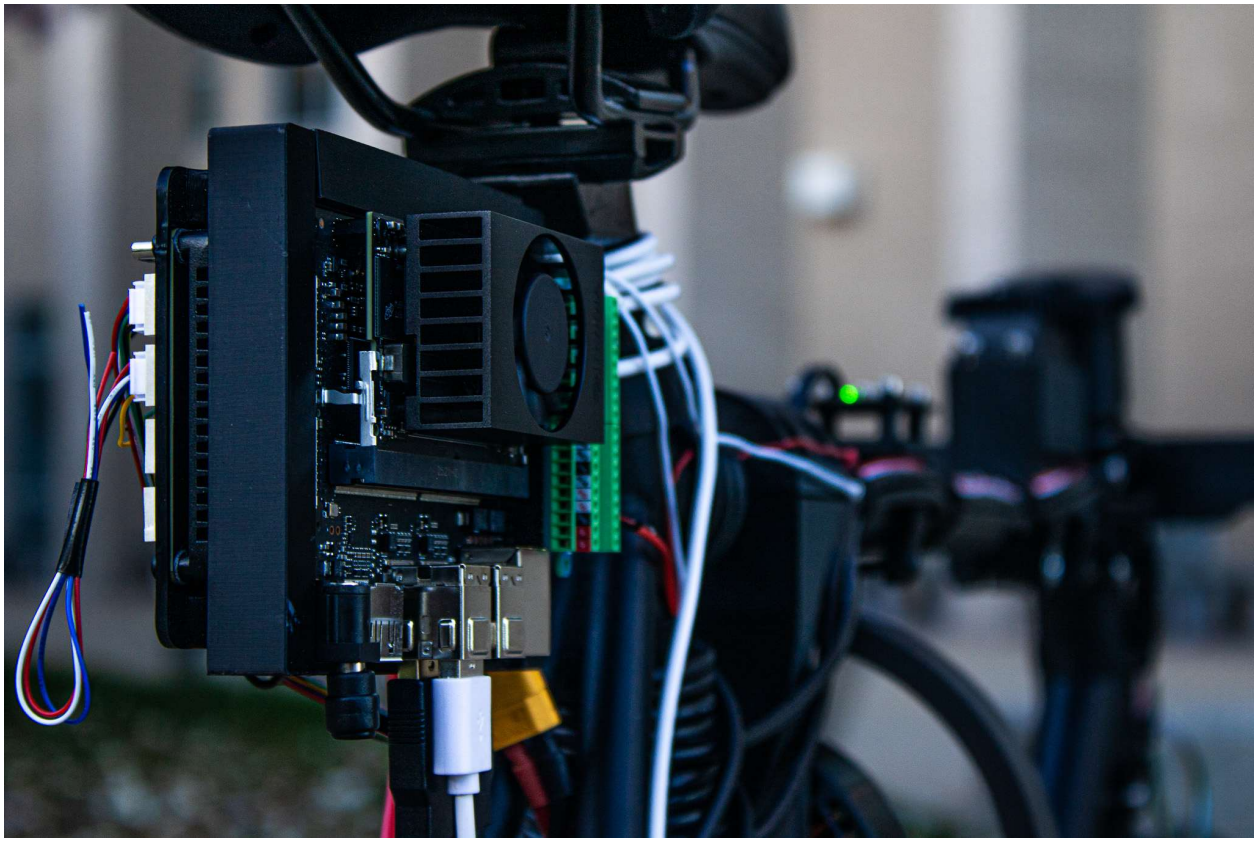
On startup, two calibration routines run before the control loop begins.
IMU_CalibrateGyro() collects 500 gyroscope samples at rest and averages them to compute a static bias offset (gy_bias). This is subtracted from all subsequent gyro readings to eliminate the zero-rate error.
IMU_InitAngle() collects 500 accelerometer samples and averages the computed atan2(ax, az) to initialize the starting tilt angle. This gives the complementary filter a physically meaningful starting condition rather than zero.

The primary balance loop is a PD controller. The integral term (Ki) is set to zero, its role is filled by the adaptive outer loop described below. Output is clamped to ±7.0 Nm and sent to the ODESC as a torque command. If the tilt angle exceeds the fall threshold (FALL_LIMIT = 0.1 rad = 5.7 degrees), the controller sends a zero-torque command and resets all state. This prevents runaway torque commands when the bike has already fallen beyond recovery.

During early tuning, a pure PID or PD loop caused any persistent lean to command the controller to spin the reaction wheel continuously in one direction. The wheel eventually reaches its maximum RPM (saturation), at which point there is no remaining torque headroom to respond to

further disturbances. The solution is an adaptive outer loop that runs every control tick. It accumulates torque commands over the interval and uses that to incrementally shift the setpoint. If the controller is consistently commanding torque in one direction, the bike's natural equilibrium is offset from the current setpoint. Nudging the setpoint toward the direction of persistent torque demand corrects that offset, which causes the mean torque demand to drop and makes the reaction wheel's angular velocity to move towards zero.

Drive & Steering Integration

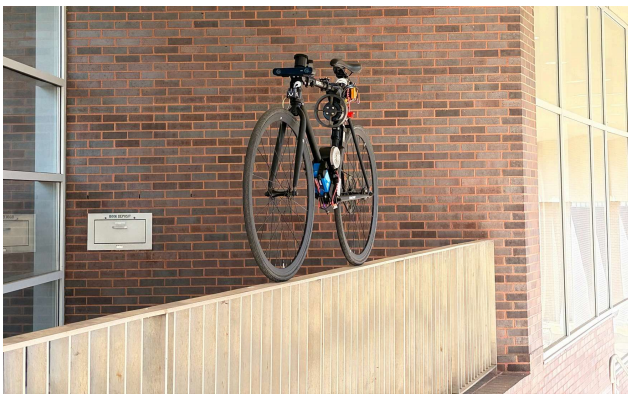


The initial approach was to generate PWM for the drive ESC and steering servo directly from the Jetson Orin Nano's GPIO. This did not work because an OS-level bug on the Jetson prevented the PWM frequency from being set below 195Hz. Both the drive ESC and steering servo are standard RC devices expecting a 50Hz signal, and they do not function correctly at higher frequencies. Both PWM signals were rerouted through the STM32, and the Jetson communicates target values over USB serial using a simple command protocol.

When steering was first tested with the active balancing, the bike fell immediately upon any steering input. Turning the front wheel shifts the natural balance point of the system. A sudden large change in the balance point produces an error spike that the reaction wheel cannot recover from quickly enough.

After measurement and curve fitting, I found a fourth-degree polynomial that maps the servo pulse width to the required setpoint offset. The polynomial is evaluated relative to the neutral position (1500us pulse width), so servo_poly() returns a signed offset from center. The offset is applied to the setpoint in set_servo_pwm() each time a new steering command arrives. This compensation updates the balance setpoint simultaneously with every steering change, so the reaction wheel always knows the new target balance point. Without it, motion of the bicycle was not achievable.

Results




The bike maintains balance continuously for as long as the battery is alive, under normal indoor and outdoor conditions without significant wind. The dynamic setpoint adjustment successfully prevents reaction wheel saturation during sustained operation. The bike completed a full remotely controlled lap around the O, validating that the balance, steering, and drive systems function together in a real outdoor environment with relatively flat surface. This demonstrated that the steering compensation polynomial and the reaction wheel stabilization are effective during motion. When subjected to external pushes, the PD loop and setpoint adjustment successfully counteracts the force, returning reaction wheel angular velocity toward zero while maintaining balance.

Future Work

The most immediate next step is deploying a full ROS2 autonomy stack on the Jetson Orin Nano. The sensing hardware is already in place, which means the remaining work is integrating visual SLAM (using Isaac ROS for best performance) and autonomous navigation (either using frontier-based exploration or waypoints). The completed ROS2 stack will be released as part of this repository.

A long term direction is replacing the current reaction wheel balance control with RL policies. An RL agent trained across a wider distribution of scenarios could produce more robust and generalizable behavior. A USD (Universal Scene Description) file of the bicycle has already been tested, making it compatible with NVIDIA Isaac Sim and other physics simulators for

sim-to-real transfer and training. The current plan is to use the reaction wheel for low speed balancing and have the RL model take over at higher speeds.

Adding an encoder to the rear wheel and steering servo would improve localization accuracy for SLAM. The current electronics also have no water or dust protection, which limits reliable outdoor operation in harsh weather conditions.

Personal Reflection

This project is probably the most technically demanding thing I have worked on, and also one of the most enjoyable. An autonomous bicycle is a true integration challenge, mechanical, electrical, embedded, control theory, and software all have to work together at once, and I don’t think a single part of it worked on the first try. We constantly iterated through different designs and components and found ways to make it work.

A lot of the learning came from failures. The NEMA24 motor that lacked the torque for balance control, the XY-BLDC controller that could not do instant direction reversals, the Jetson PWM frequency limitation, and many more. Each of these required diagnosing the root cause, debugging, and applying a fix. The team did really well at pivoting and finding new approaches rather than getting stuck. That willingness to throw out a design that wasn't working, even when we had worked on it for weeks, is probably the biggest reason this project came through in the end.