GYROBOT

(Self-balancing robot)

Introduction:

Autonomous systems such as drones or self-balancing mobile robots are playing an increasingly important role in industrial and societal applications. One of the key principles these systems rely on is the ability to stabilize inherently unstable systems in real time by using inertial sensors and control algorithms.

During my second year at ECE, I participated in the DrawBot project, which involved building a robot using the electronic platform provided by the school. I handled the hardware setup (soldering, wiring, assembling the motors, wheels, and the ESP32 microcontroller) and then programmed the system so that it could draw various shapes, such as a rosette, a compass rose oriented to magnetic North, or a right-angled staircase.

Wanting to go beyond this academic framework and improve my skills in PID control, inertial sensor usage, and data processing through filtering, I challenged myself to repurpose the DrawBot hardware structure to design a GyroBot: a balancing robot capable of continuously stabilizing itself around its equilibrium position.

Photo of the DrawBot

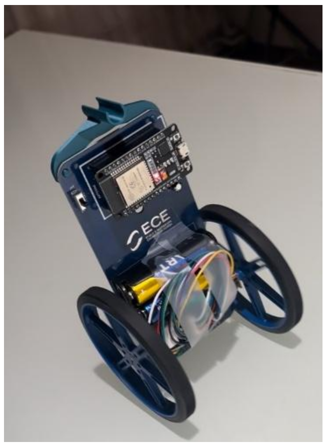


Photo of the GyroBot

Hardware used :

Microcontroller - Dual-core NodeMCU ESP32 running at 240 MHz, featuring 512 kB of SRAM and 4 MB of Flash memory. It handles sensor acquisition, PID control computation, and motor driving.

Inertial Measurement Unit (IMU) - LSM6DS3 (6-axis IMU) Sensor integrating a triaxial accelerometer and a triaxial gyroscope:

1. The accelerometer provides a reliable long-term estimate of the angle, but it is noisy in the short term.
2. The gyroscope provides high instantaneous accuracy but exhibits cumulative drift over time.

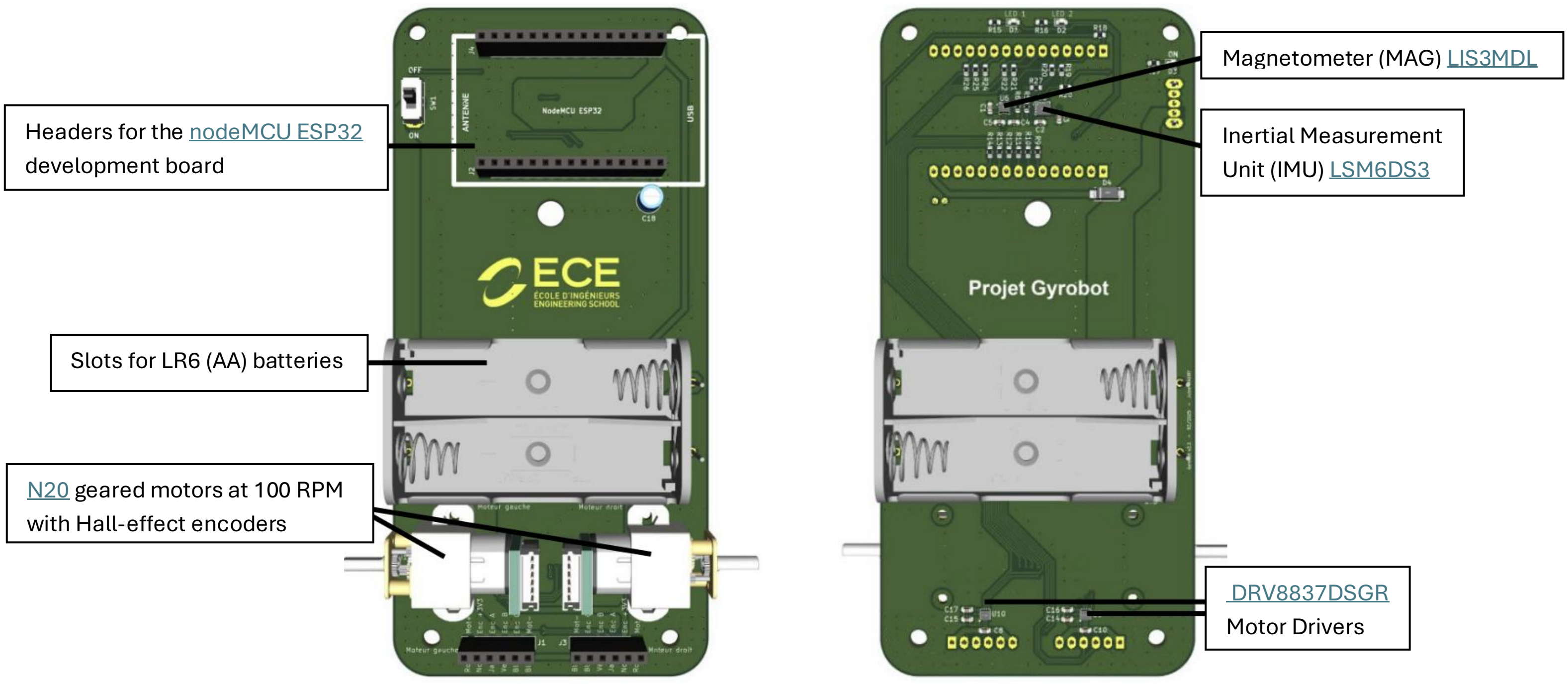
Combining the two through a complementary filter provides a robust and responsive angle, essential for the robot's balance.

Motorization - N20 geared motors with integrated encoders: two N20-type DC motors, each equipped with a Hall-effect encoder to measure wheel rotation (useful for precise control of movement, though not essential for this project). The gearboxes limit the speed to about 100 RPM, which increases the available torque and allows the robot to move despite the low nominal power. However, this constraint requires very fine error margins for control.

Since these geared motors come from a previous project, mechanical play due to wear is present in the wheels, introducing additional disturbances. These must be compensated by the control system to ensure stable movement.

Power Supply – Two LR6 (AA) battery packs. The power supply is distributed symmetrically on either side of the board, which helps balance the weight and minimizes the chassis's initial imbalance. The delivered voltage (≈ 6 V) is regulated to power the motors and onboard electronics.

The robot's geometry, originally designed for a DrawBot project, is not suited for a self-balancing robot, which increases the difficulty of stabilization.


Headers for the nodeMCU ESP32 development board
Slots for LR6 (AA) batteries
N20 geared motors at 100 RPM with Hall-effect encoders
ECE
Projet Gyrobot
Magnetometer (MAG) LIS3MDL
Inertial Measurement Unit (IMU) LSM6DS3
DRV8837DSGR Motor Drivers

Methodology :

1 – Sensor Data Acquisition and Processing

To determine the GyroBot's angle, two sensors from the IMU were used:

- Gyroscope: The angle is obtained by the temporal integration of the angular velocity. However, this sensor has an offset that causes a gradual drift of the angle over time: the longer the integration, the more the error accumulates.
- Accelerometer: The angle is estimated from the distribution of gravity along its axes. This measurement is reliable in the long term but highly noisy, with fluctuations of up to $\pm 5°$ even when the system is stationary.

Thus :

- The gyroscope is accurate in the short term but unstable in the long term (drift).
- The accelerometer is somewhat usable in the short term (noisy) but precise in the long term.

Adopted solution: A complementary filter was implemented, which combines:

- The gyroscope's instantaneous precision
- The accelerometer's long-term stability

Result: A very accurate angle, with variation reduced to $\pm 0.02°$ at static equilibrium.
A quick analysis confirmed that the roll angle (X-axis) was the relevant degree of freedom for balancing, becoming the variable to control.
To formalize this fusion principle, the estimated angle is calculated using the following complementary filter:

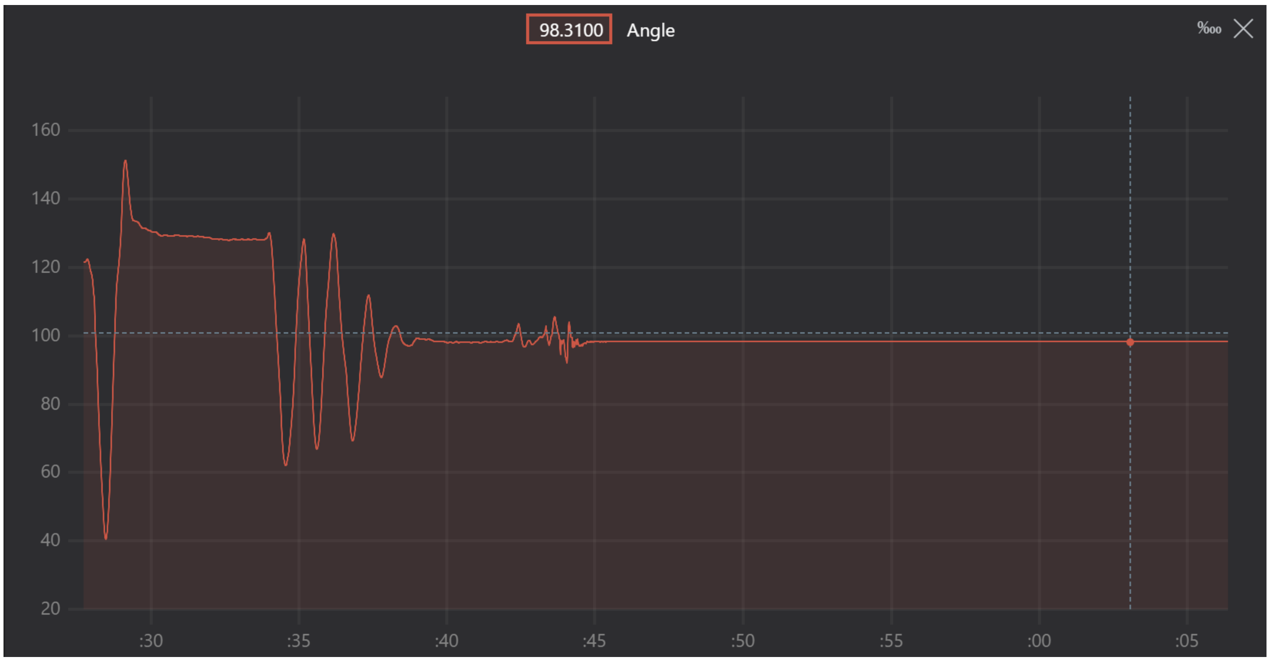
$$\theta(t) = \alpha.\left(\theta(t-1) + \omega_{gyro}(t)\Delta t\right) + (1-\alpha).\theta_{acc}(t)$$

α : gyroscope weight
ω_{gyro} : angular velocity measured by the gyroscope
θ_{acc} : angle calculated by the accelerometer

2 – Determination of the equilibrium angle
Problem:
The robot's mechanical structure is not perfectly suited for a self-balancing robot, so the equilibrium position does not correspond to a 90° angle (ideal vertical). It was therefore necessary to carry out a specific step to experimentally identify the actual equilibrium angle.

Methodology:
The Teleplot tool integrated into Visual Studio Code was used to visualize the angle measured by the IMU in real time. The robot was then manually stabilized several times until the position where it could maintain itself the longest without external intervention was identified.

Result:
The robot's equilibrium angle was experimentally determined to be 98.3°, which was adopted as the reference setpoint for the control system.



Graphical representation showing the manual search for the GyroBot's equilibrium position

3 – PWM Signal Generation for Motor Control
Problem:
To control the wheels' rotation speed, it is necessary to apply a variable voltage to the motors. However, the ESP32 does not have analog outputs, only digital outputs (HIGH/LOW). Without an appropriate solution, the microcontroller could only provide two states to the motors (stop or full speed), making fine speed regulation impossible.

Solution:
The ESP32 includes 16 PWM (Pulse Width Modulation) channels. This technique consists of rapidly oscillating a digital output at a given frequency (here 1 kHz) while modulating the duty cycle (ratio of HIGH to LOW). The average value of this signal is perceived by the motors as an intermediate voltage, which allows simulating an analog output and precisely controlling their speed.

Result:
Using PWM enables continuous modulation of the wheels' speed from the ESP32's digital outputs. This principle can be expressed mathematically by the following relation:

$$V_m = \frac{t_{high}}{T_{cycle}}.V_{cc}$$

4 – PID Control Loop
Problem:
The main objective of the project is to stabilize the robot around its equilibrium angle. To achieve this, a closed-loop control system is required. However, a complete system model (transfer function) was not available, making a theoretical PID controller tuning impossible.

PID Methodology:
The PID controller applied to stabilize the GyroBot is expressed by the following equation:

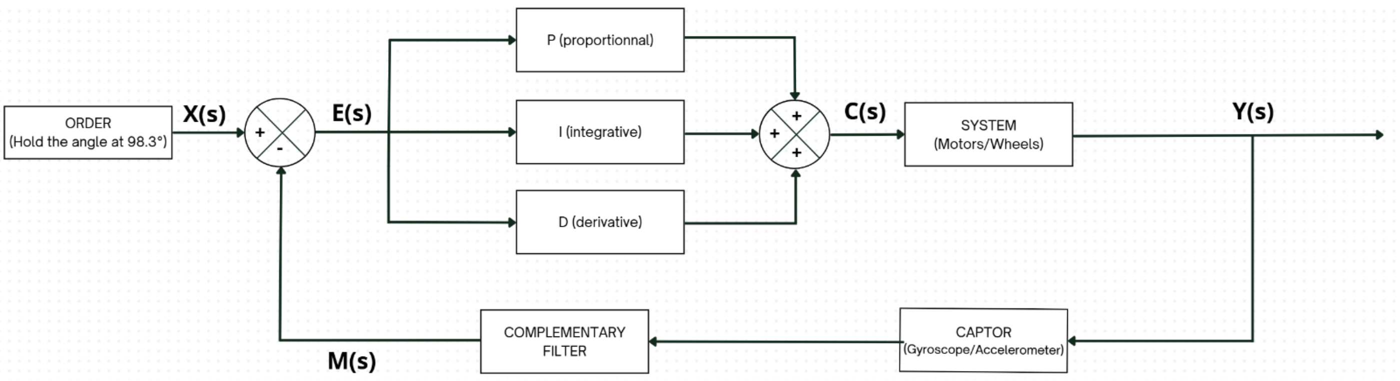
$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}$$

with $e(t) = \theta_{ref} - \theta(t)$

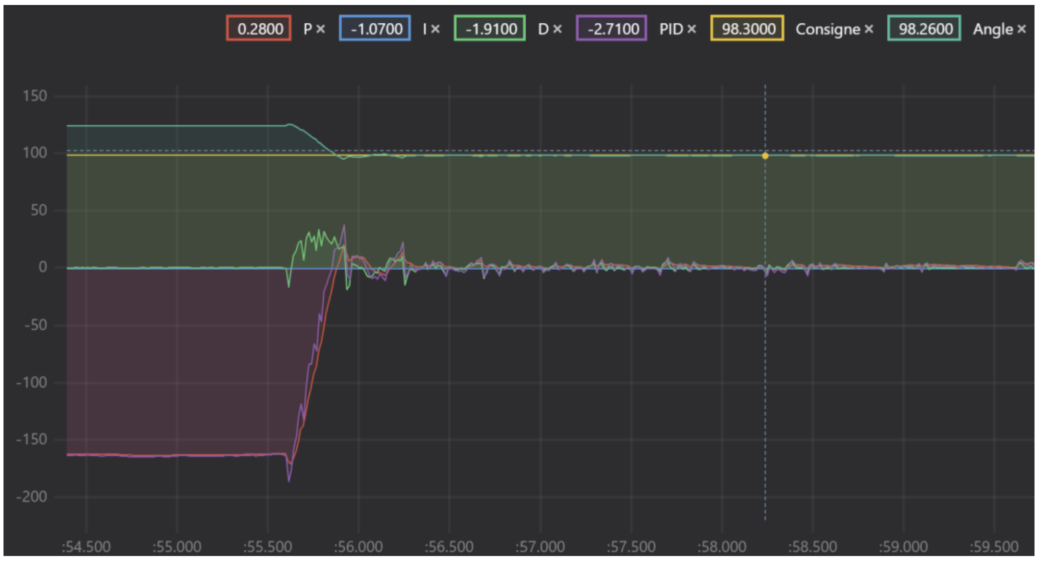
The PID tuning was carried out empirically, through successive trials on the coefficients Kp, Ki, and Kd:

1. Tuning the proportional (P) controller:
   The Kp term directly converts the angle error into motor commands. It was adjusted first (with Ki = Kd = 0). An appropriate value allows the robot to stand upright but still causes oscillations.
2. Tuning the derivative (D) controller:
   The Kd term acts as a damper by limiting rapid angle changes. By adjusting it (with Ki = 0), the oscillations were reduced, enabling the robot to balance for short periods.
3. Tuning the integral (I) controller:
   Finally, Ki was introduced to correct the residual error that remained over the long term and prevented permanent balance. This term compensates for small mechanical, or sensor biases and ensures continuous stability of the robot.

Result:
Thanks to this incremental tuning procedure, the robot can maintain its equilibrium position autonomously and sustainably, with a stable response and no significant long-term drift.



Block diagram showing the closed-loop control operation of the GyroBot



Graphical representation of the GyroBot balancing procedure showing the response of all PID controllers and the angle relative to the setpoint

Possible improvements :

Mecanichal improvements

- Design a dedicated chassis for the balancing robot with a symmetrical geometry and balanced mass distribution (batteries and electronics positioned symmetrically) to lower the center of gravity and reduce unwanted vibrations.
- Reduce mechanical play on the shafts (adjust bearings, use shims, select appropriate bearings).
- Increase the power of the gear motors with a better power supply and improve wheel grip/quality for a more responsive movement.

Control optimization

- Develop a simplified differential model (small angles) and use Scilab to fine-tune the Kp, Ki, and Kd gains before real-world testing.
- Implement a Kalman filter for a more robust angle estimation than a simple complementary filter, especially to handle noise and drift.

Advanced features

- Integrate a Bluetooth/Wi-Fi interface to monitor the angle and adjust PID gains in real time (telemetry and remote tuning).

Conclusion :

This project demonstrated the successful transformation of a platform initially designed as a DrawBot into a self-balancing GyroBot, capable of stabilizing around its equilibrium point. By utilizing inertial sensors, applying a complementary filter for precise angle estimation, and tuning a PID controller, the robot was able to maintain its balance despite mechanical constraints such as limited motor power and structural play.

- Sensor data processing and fusion, through the implementation of a complementary filter to estimate the tilt angle.
- Real-time programming and hardware control, notably by manipulating PWM signals to precisely control motor speed.
- Iterative experimental approach, using tools like Teleplot to identify and validate the system’s behavior.
- Application of control theory, with the empirical tuning of a PID controller.

The GyroBot thus served as a practical laboratory for experimenting with the stabilization of a naturally unstable system and provided a solid foundation for more advanced robotics projects. Future developments will focus on a mechanically optimized design, the use of more advanced filters, and the integration of remote monitoring and control features.

Ultimately, this project represents a key step in my robotics journey and directly prepares for the development of more complex systems such as autonomous drones, where multi-axis stabilization, sensor fusion, and trajectory planning play a central role.

DOC EXTERNE :

Vidéo fonctionnelle : https://youtu.be/EcyiQm1V0ns

GitHub: https://github.com/sachacogo/GyroBot