Two Wheels Self-balancing Robot

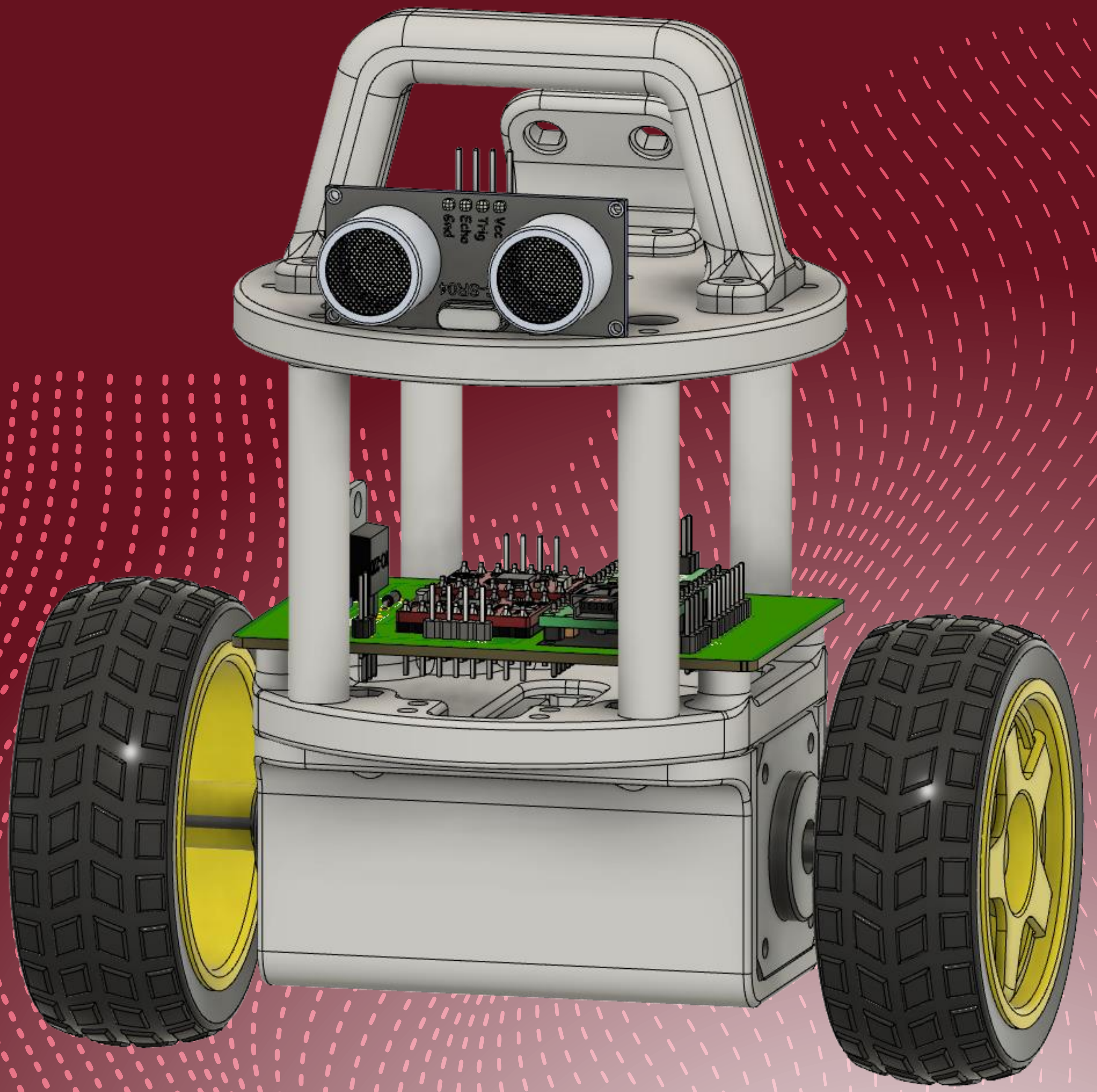
Digital Control Systems

Students: Simone Orelli, Antonio Rapuano, Francesco Russo.

DEPARTMENT OF COMUPUTER, CONTROL AND MANAGMENT ENGINEERING ANTONIO RUBERTI

Professor: Claudia Califano

SAPIENZA
UNIVERSITÀ DI ROMA



Outline

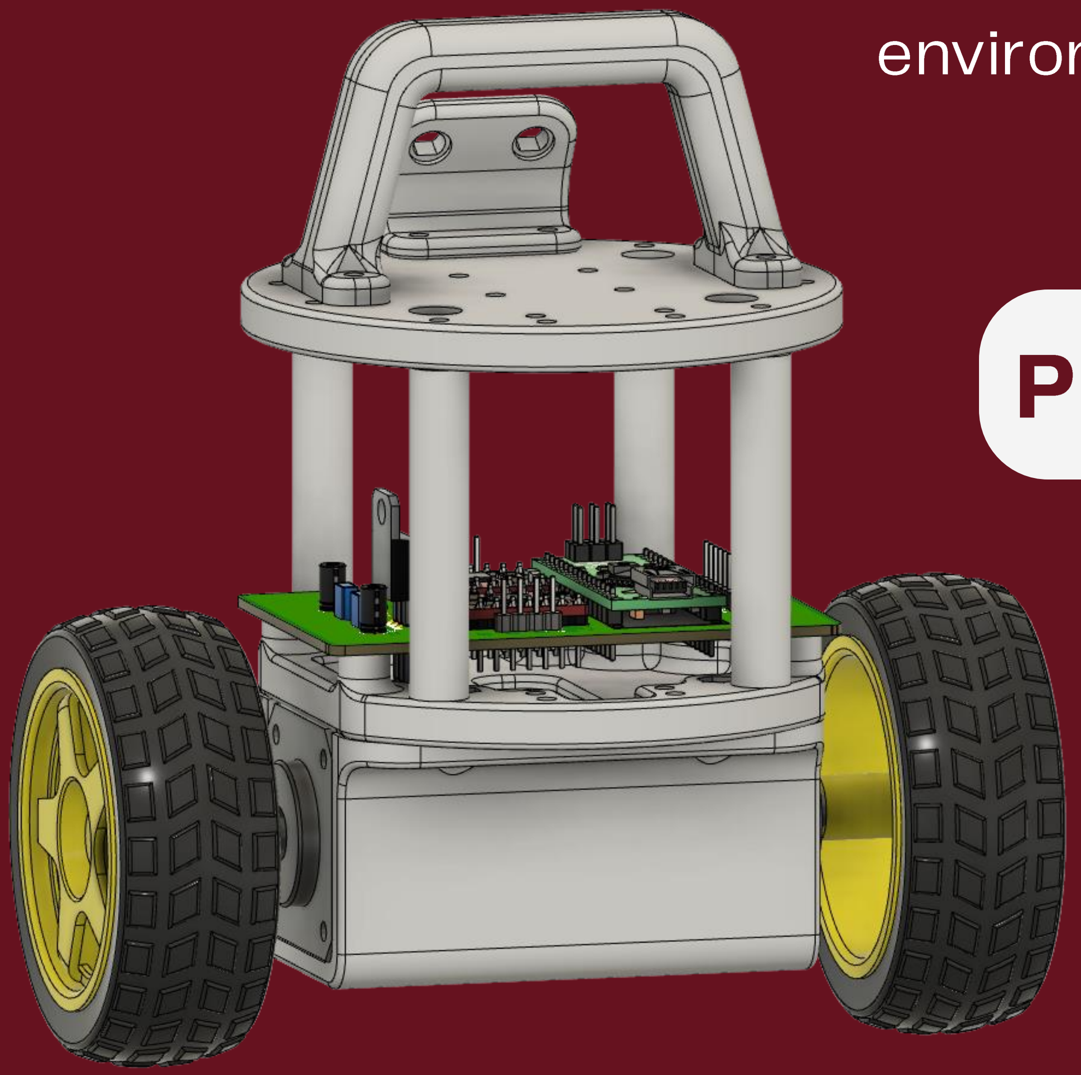
SAPIENZA
UNIVERSITÀ DI ROMA

- Introduction
- Hardware
- Model
- Simulation
- Sensor Data Acquisition
- Demo
- Future Developments
- Conclusion

Introduction

Goal

- The aim of this project is to build a wheeled robot with the remarkable capability of autonomously maintaining balance.
- The heart of this project is the ambition to empower the robot with the ability to dynamically adapt to variations in its surrounding environment.



Physical Structure

- The physical structure of the robot has been designed using **Autodesk Inventor** and subsequently manufactured with a 3D printer.
- The material chosen to print the robot is PLA, which offers good strength and rigidity.

STVDIVM VRBIS
SAPIENZA
UNIVERSITÀ DI ROMA

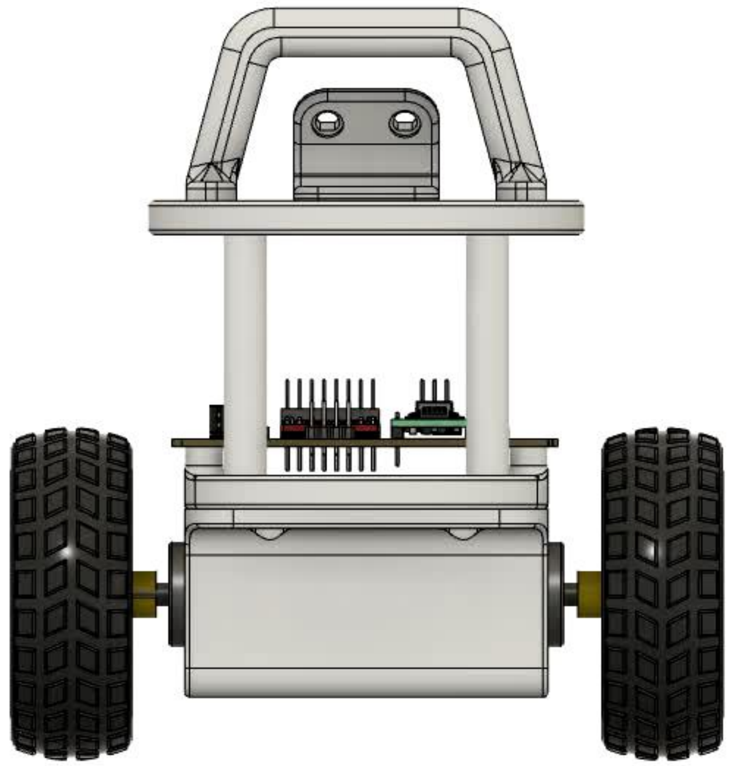
Hardware

SAPIENZA
UNIVERSITÀ DI ROMA

Two Wheels Self Balancing Robot 3D Model

Actuation Devices:

- Stepper Motor
  NEMA 14

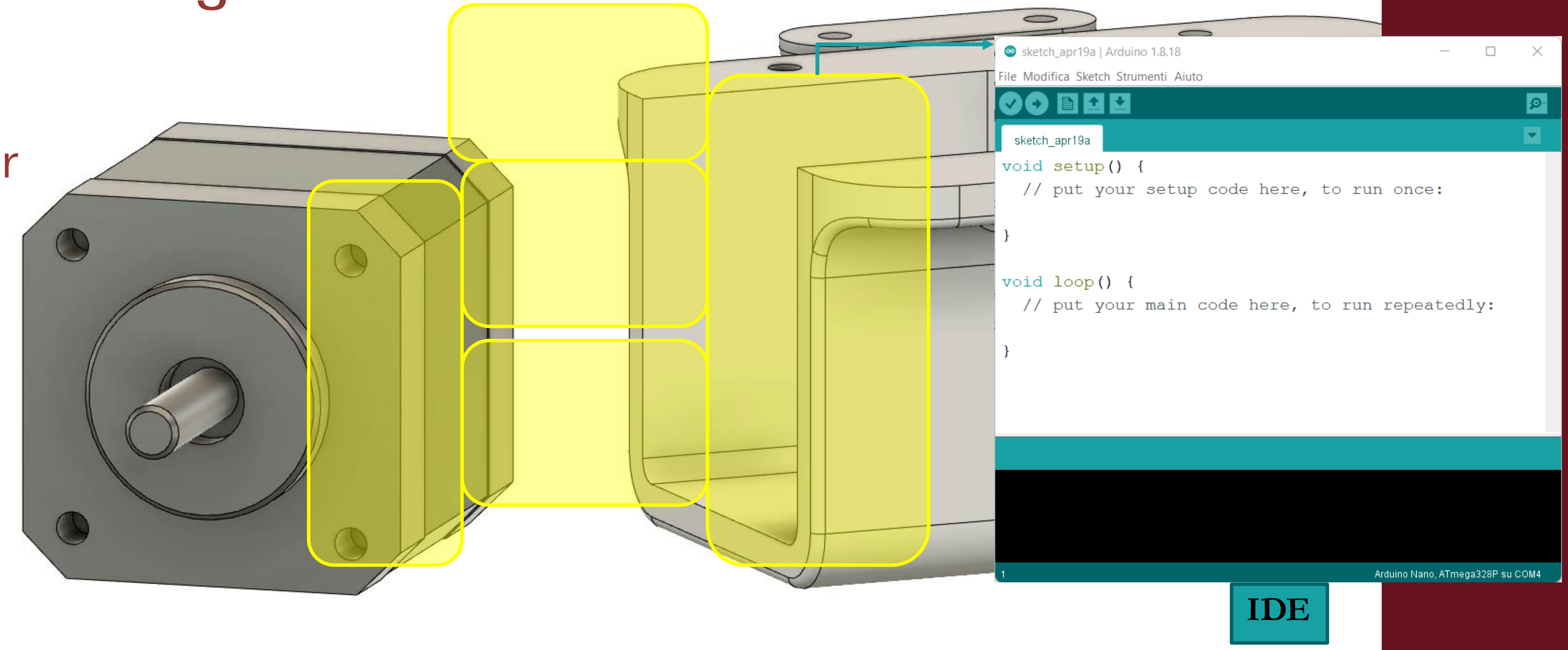


Hardware

Two Wheels Self Balancing Robot 3D Model

PCB:

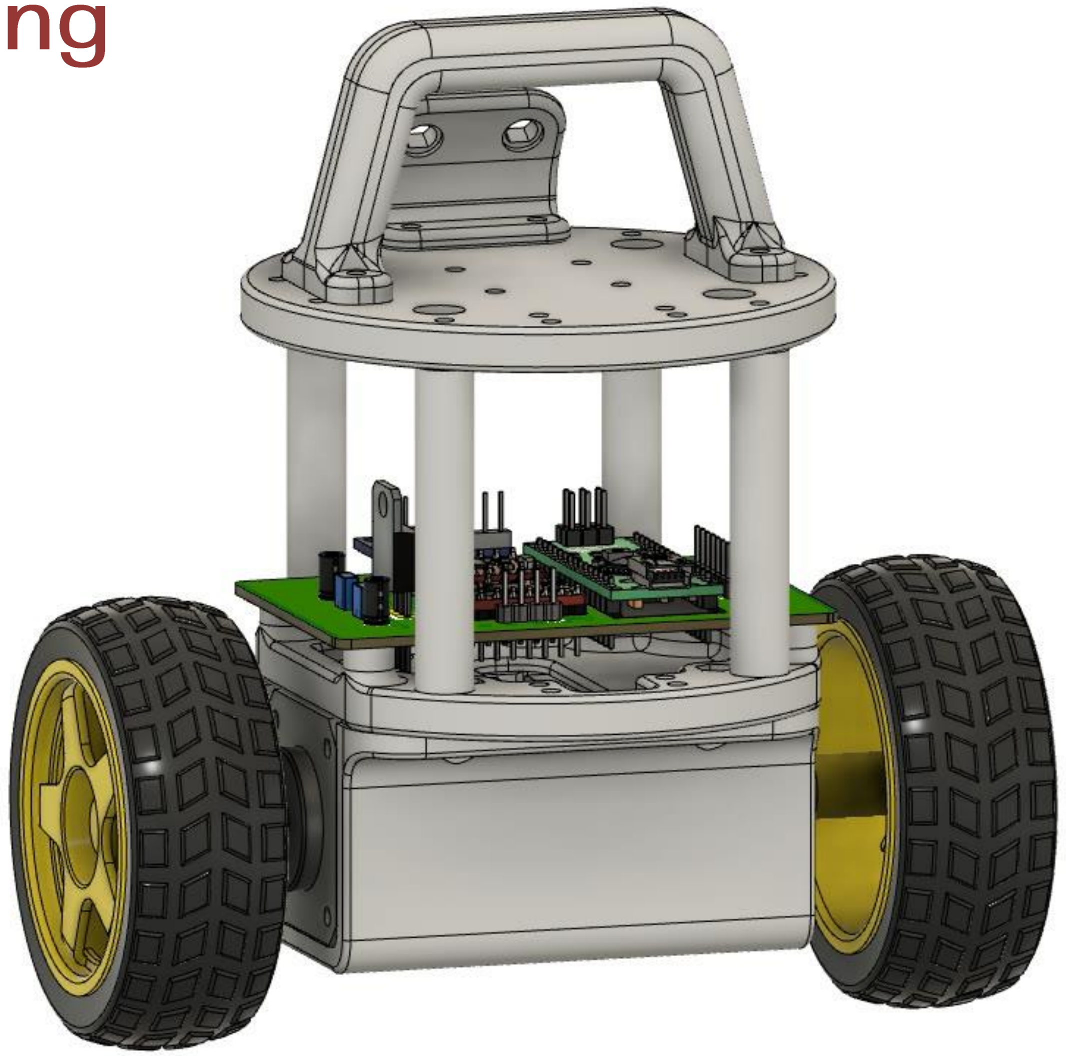
- Voltage Regulator
- Arduino Nano
- Driver A4988
- MPU-6050



Hardware

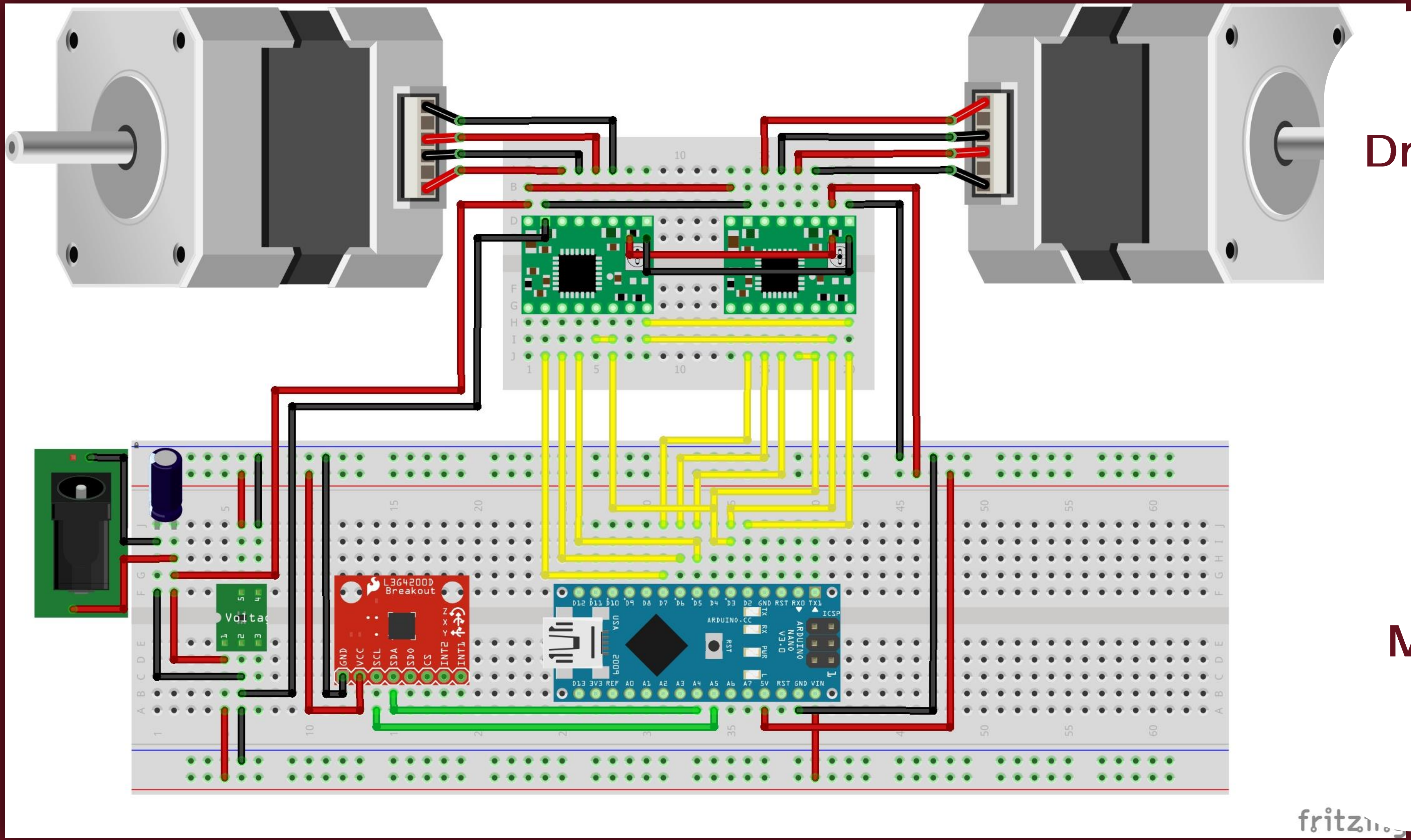
Sapienza Università di Roma

Two Wheels Self Balancing Robot 3D Model



Hardware

Electrical Scheme



CONNECTIONS

Driver A4988:

DIR –> D2
STEP –> D3
SLEEP –> D4
RESET –> D4
MS3 –> D5
MS2 –> D6
MS1 –> D7

MPU – 6050:

SCL –> A4
SDA –> A5

Model

SAPIENZA
UNIVERSITÀ DI ROMA

Lagrangian Method

- To find the dynamics of the TWSBR system we used the Lagrangian method, instead of the Newtonian one.

- The Lagrangian function is defined as:

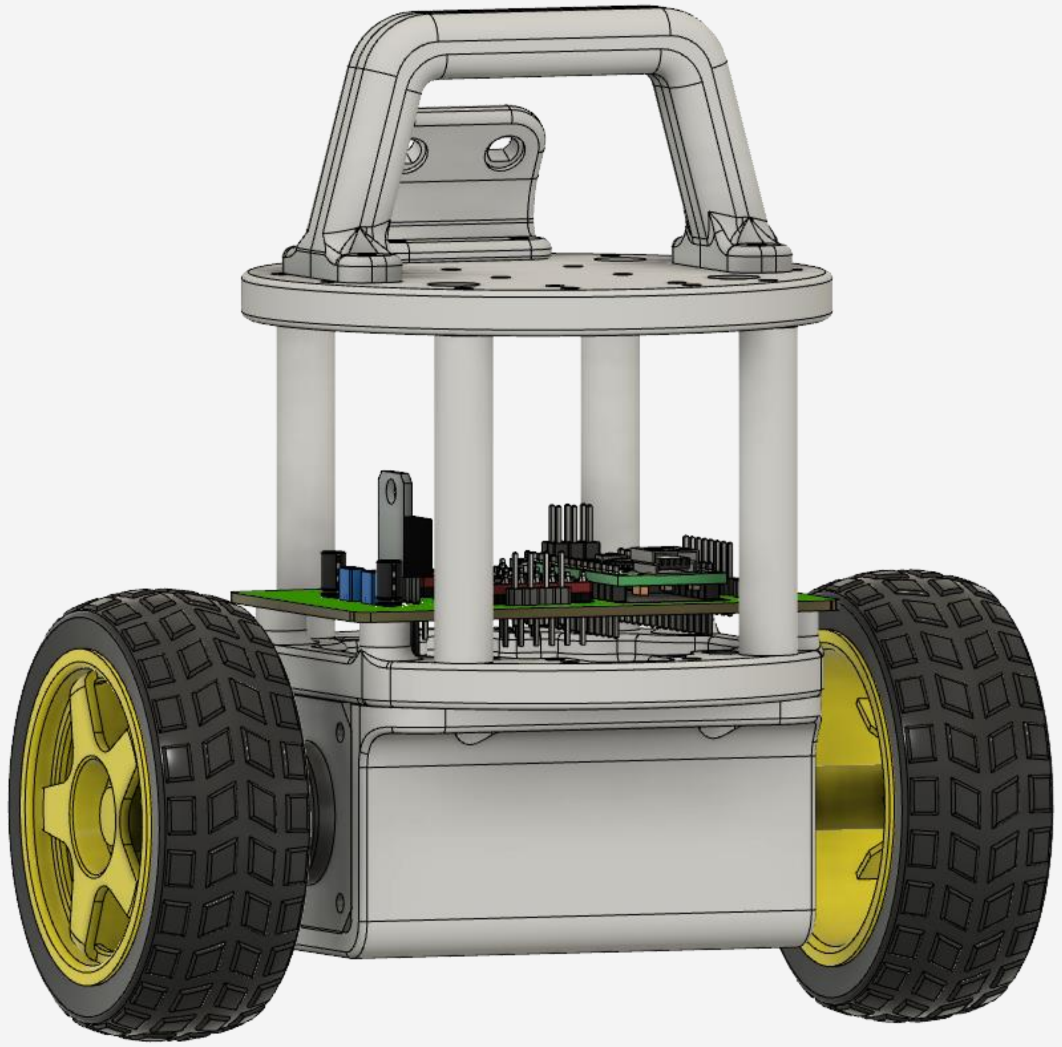
$$\mathbf{L = T - V}$$

where T is the kinetic energy and V is the potential energy.

- It is used into the Euler-Lagrange equation:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = u_i$$

where $\boldsymbol{u_i}$ are non-conservative (external or dissipative) generalized forces performing work on $\boldsymbol{q_i}$.



Model

Lagrangian Method

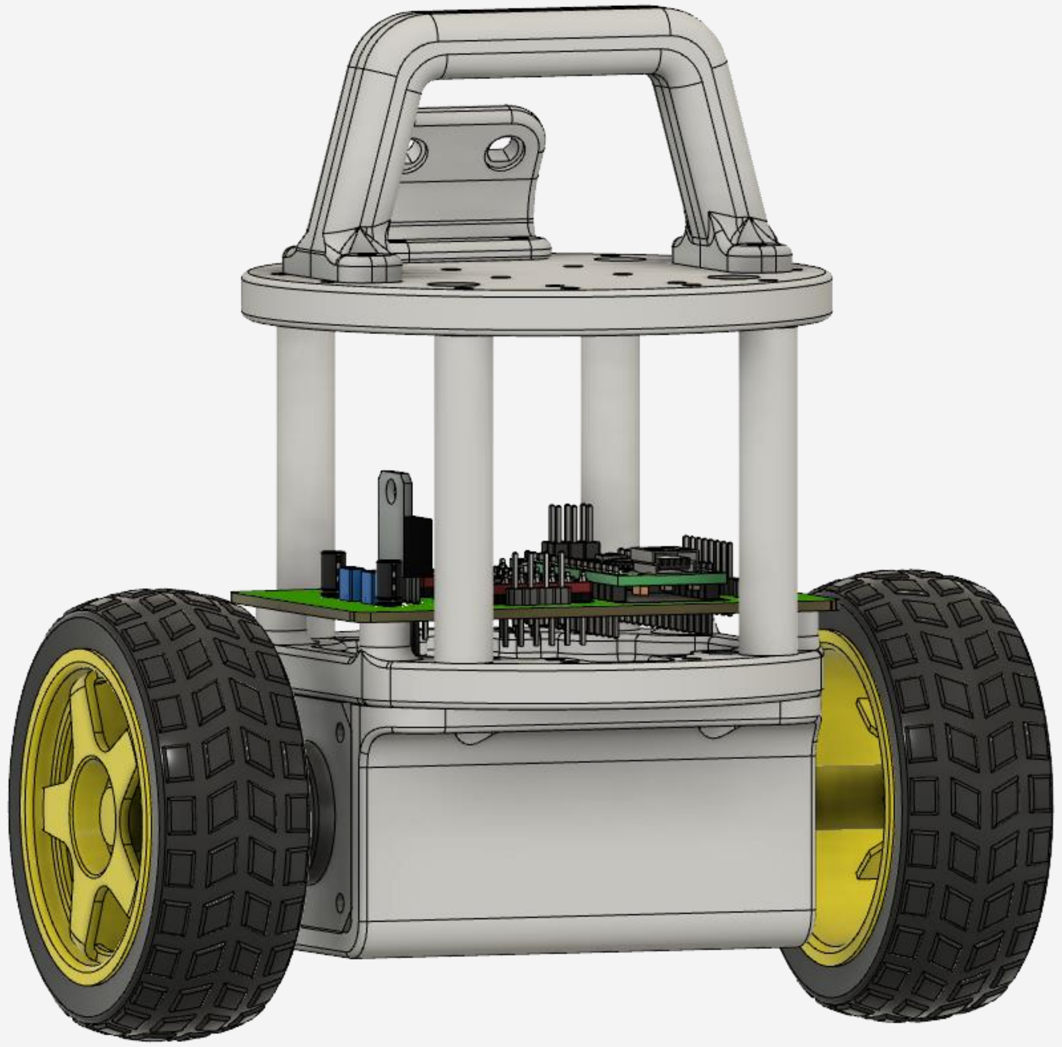
- **Euler-Lagrange equation**

$$M(q)\ddot{q} + c(q,\dot{q})\,\dot{q} + e(q) = H\tau$$

$$M(q) = \begin{bmatrix} m_b + 2m_w + \frac{2J_w}{r^2} & -m_B l_G \cos(\theta) \\ -m_b l_G \cos(\theta) & J_b + m_b l_G^2 \end{bmatrix} \qquad \mathrm{H} = \begin{bmatrix} 1 \\ \frac{-}{r} \\ 1 \end{bmatrix}$$

$$c(q,\dot{q}) = \begin{bmatrix} 0 & m_B l_G \sin(\theta)\,\dot{\theta} \\ 0 & 0 \end{bmatrix} \qquad e(q) = \begin{bmatrix} 0 \\ -m_b l_G g sin(\theta) \end{bmatrix}$$

It can be linearized about $\theta = \dot{\theta} = 0$
using $\sin(\theta) \simeq \theta, \cos(\theta) \simeq 1$ and $\dot{\theta}^2 \simeq 0$



Model

Transfer Function Representation

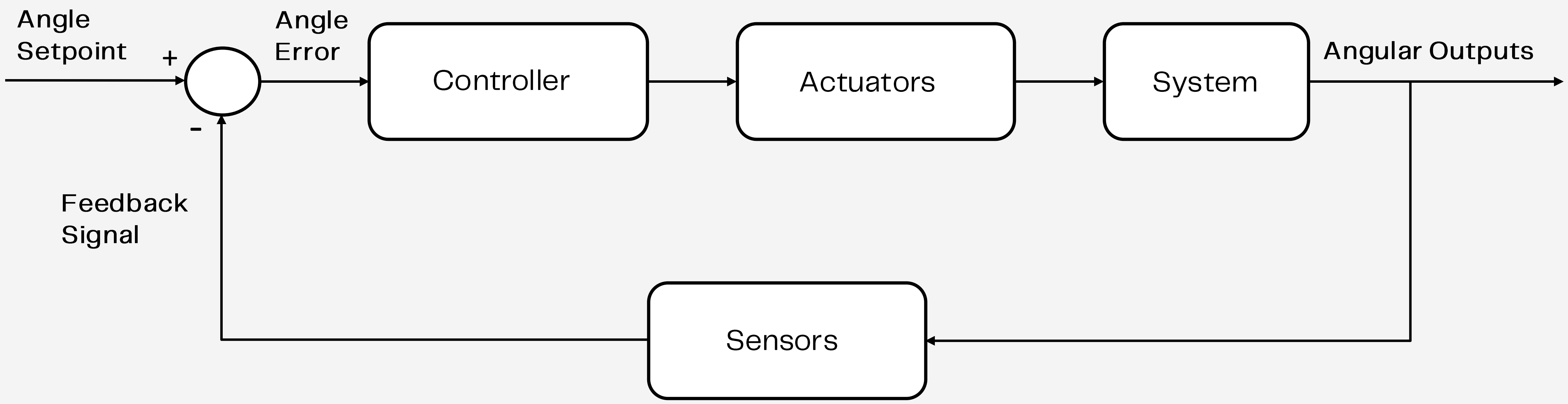
- The **stepper motors** make the system act like a **closed loop**, applying the necessary torque to reach the desired motor shaft **angular speed**.
- Because of this, there is no need to use angular encoders, but this kind of representation requires to be tuned accurately according to the motor specifications.
- **Transfer function** between pitch angle and motor input:

$$\frac{\theta(s)}{u(s)} = \frac{3203.6s}{(s+7409)(s-6.915)(s+6.9)}$$

Stability

- The TWSBR is an **unstable** system because its transfer function has an unstable pole in $p = 6.915$.

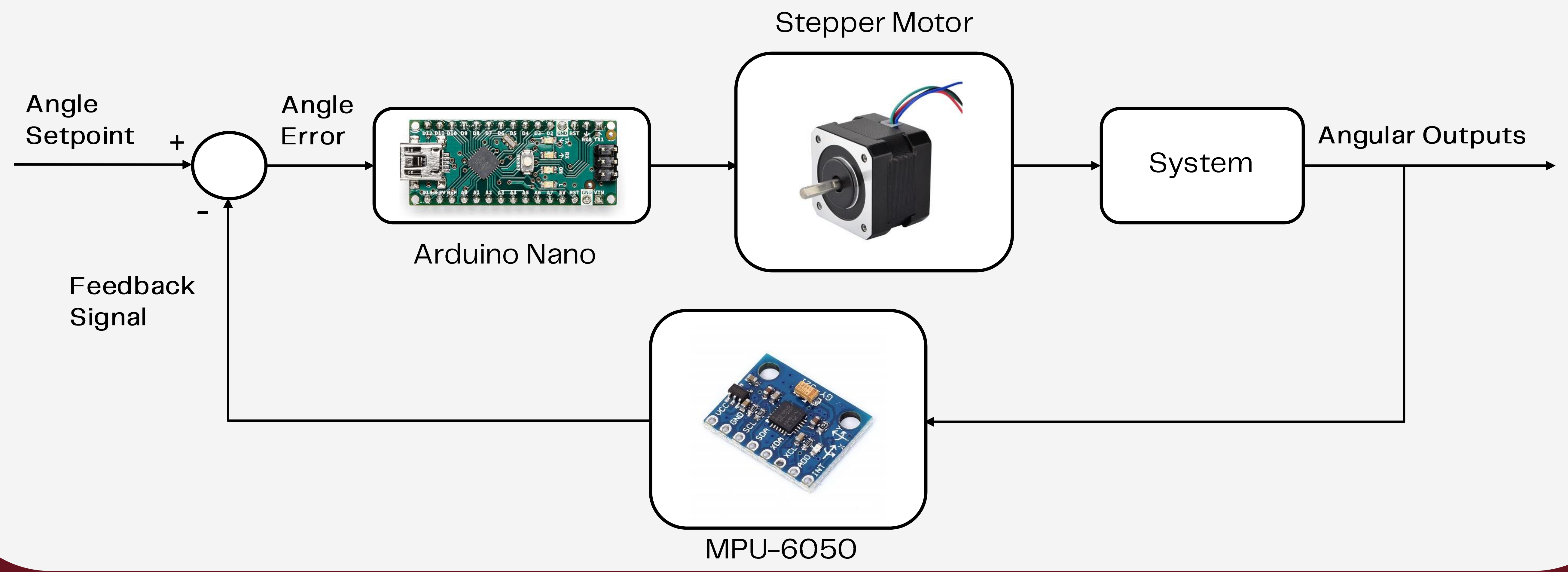
Control System



- **Feedback signals**: Angular measurements θ and $\dot{\theta}$
- **Control signal**: Angular velocity of the stepper motor shaft

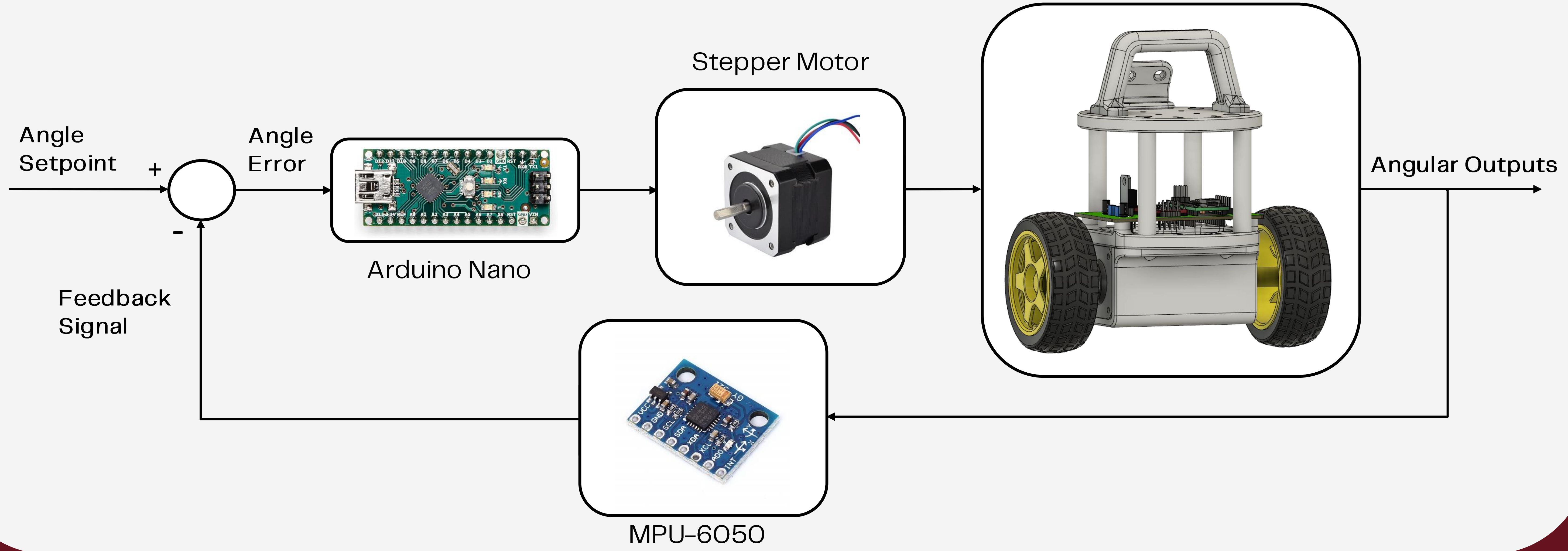
Control System

SAPIENZA
UNIVERSITÀ DI ROMA



- **Feedback signals**: Angular measurements θ and $\dot{\theta}$
- **Control signal**: Angular velocity of the stepper motor shaft

Control System

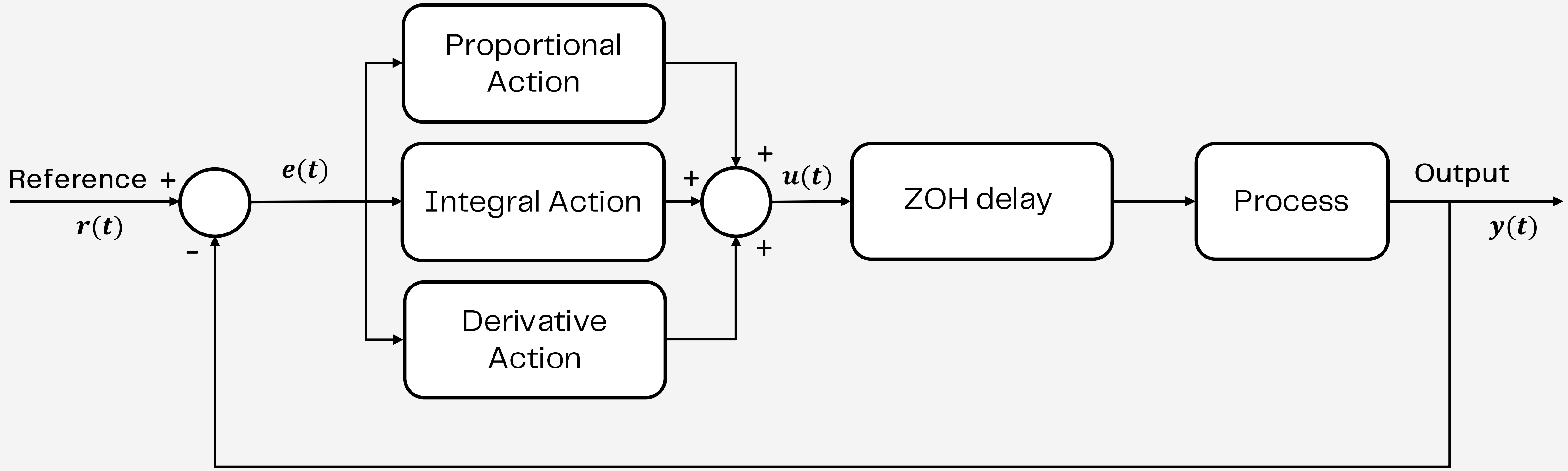


- **Feedback signals:** Angular measurements θ and $\dot{\theta}$
- **Control signal:** Angular velocity of the stepper motor shaft

Simulation

Continuous Controller

- **Continuous** representation of the discrete closed loop with the PID controller:



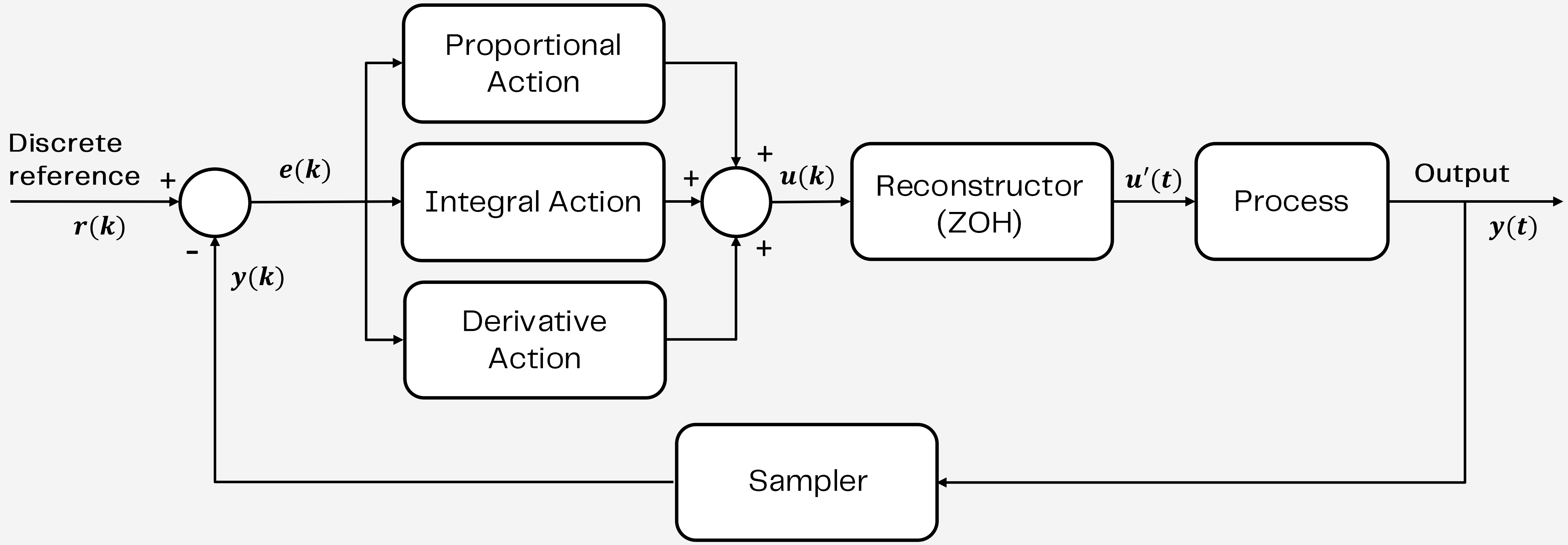
- The PID controller must take into account the **delay** introduced by the ZOH reconstruction filter, which can be approximated using Padé's formula:

$$ZOH = e^{-sT} \simeq \frac{1}{1 + \frac{sT}{2}}$$

Simulation

Discretized Controller

- The only difference lies in the way the **derivative** and the **integral** operations are computed.



- The **sampling time** is chosen sufficiently shorter than the maximum imposed by Shannon's theorem considering the bandwidth of the system.

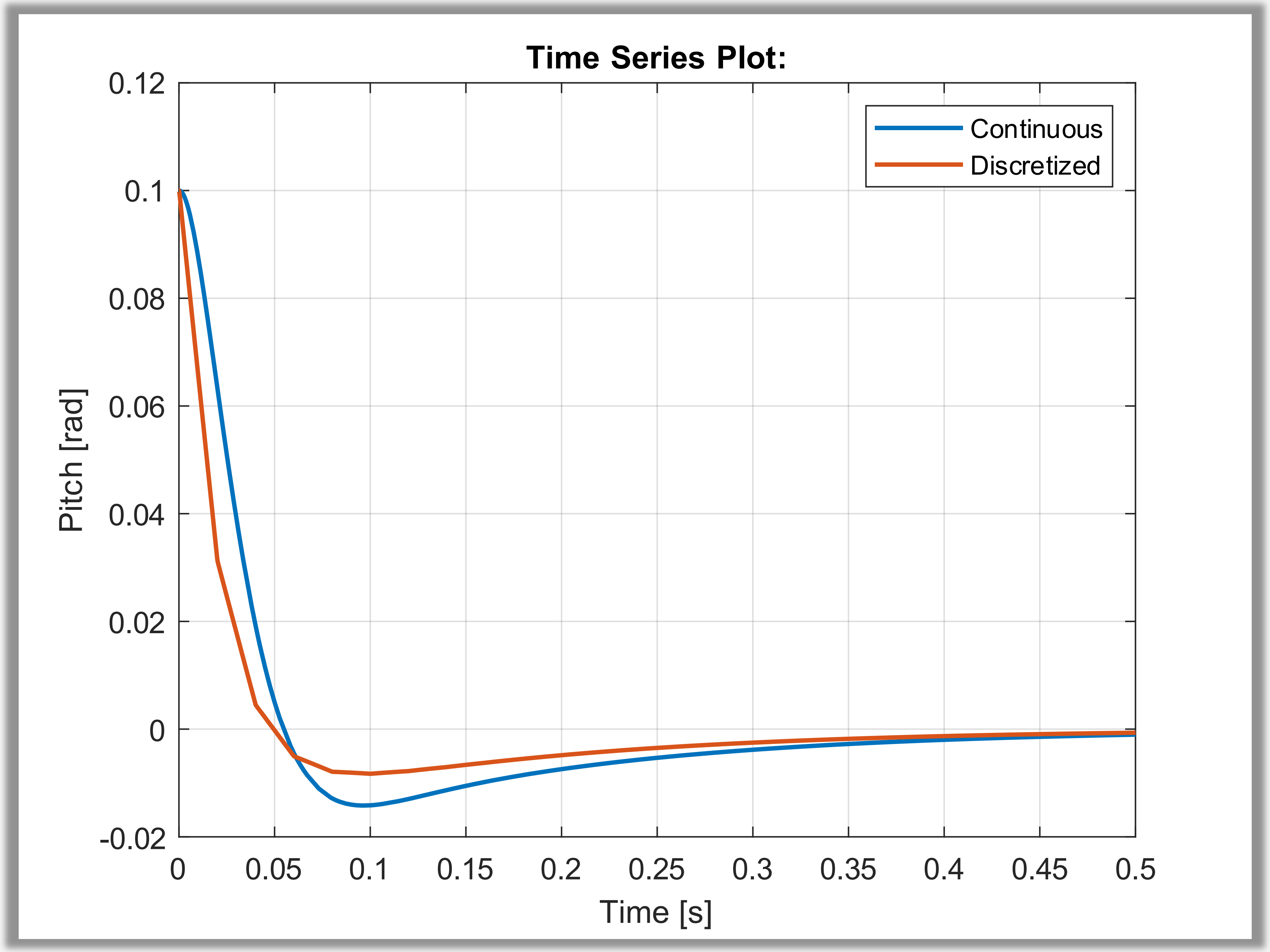
Kp = –50 Kd = –1 Ki = –375

T = 0.02 s

Simulation

SAPIENZA UNIVERSITÀ DI ROMA

- **Evolution** of the closed-loop TWSBR system with null reference.

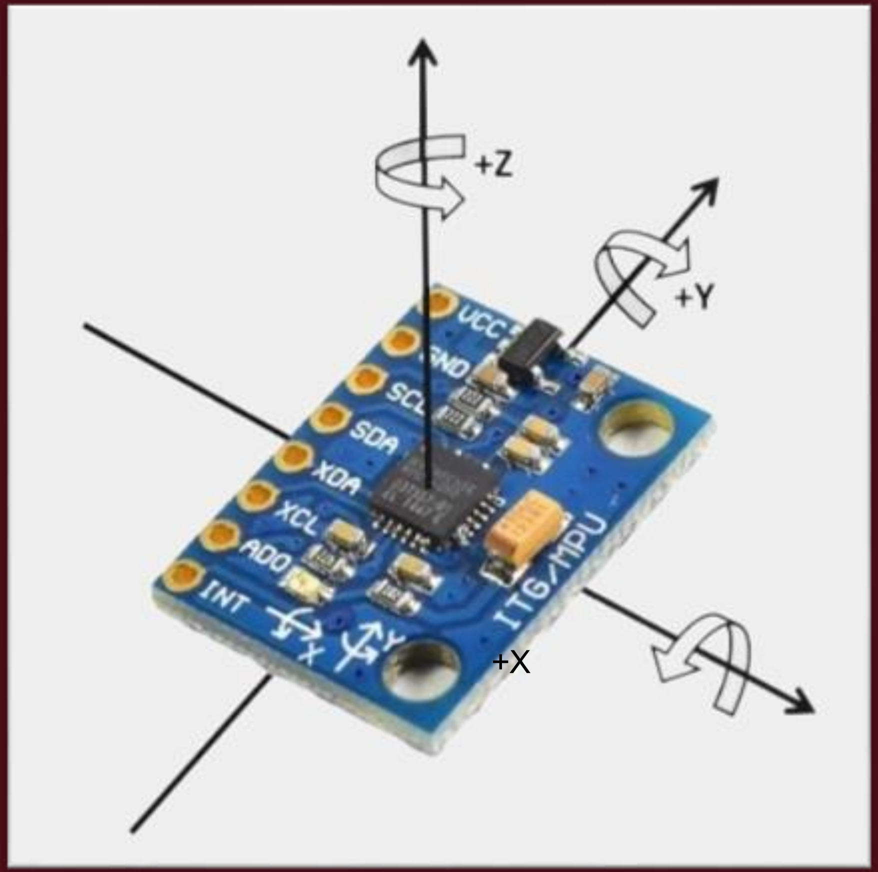


Orange line: Closed-loop response when employing the continuous version of the system.

Blue Line : Closed-loop response when using the discrete implementation of the continuous controller.

Sensor Data Acquisition

MPU6050



Gyroscope:
directly measures the angular rate $\dot{\boldsymbol{\theta}}$.

Accelerometer:
measures the gravity acceleration, which can be used to compute the angular displacement θ.

} Sensor data fusion

Sensor Data Acquisition

Significant presence of measurement noise

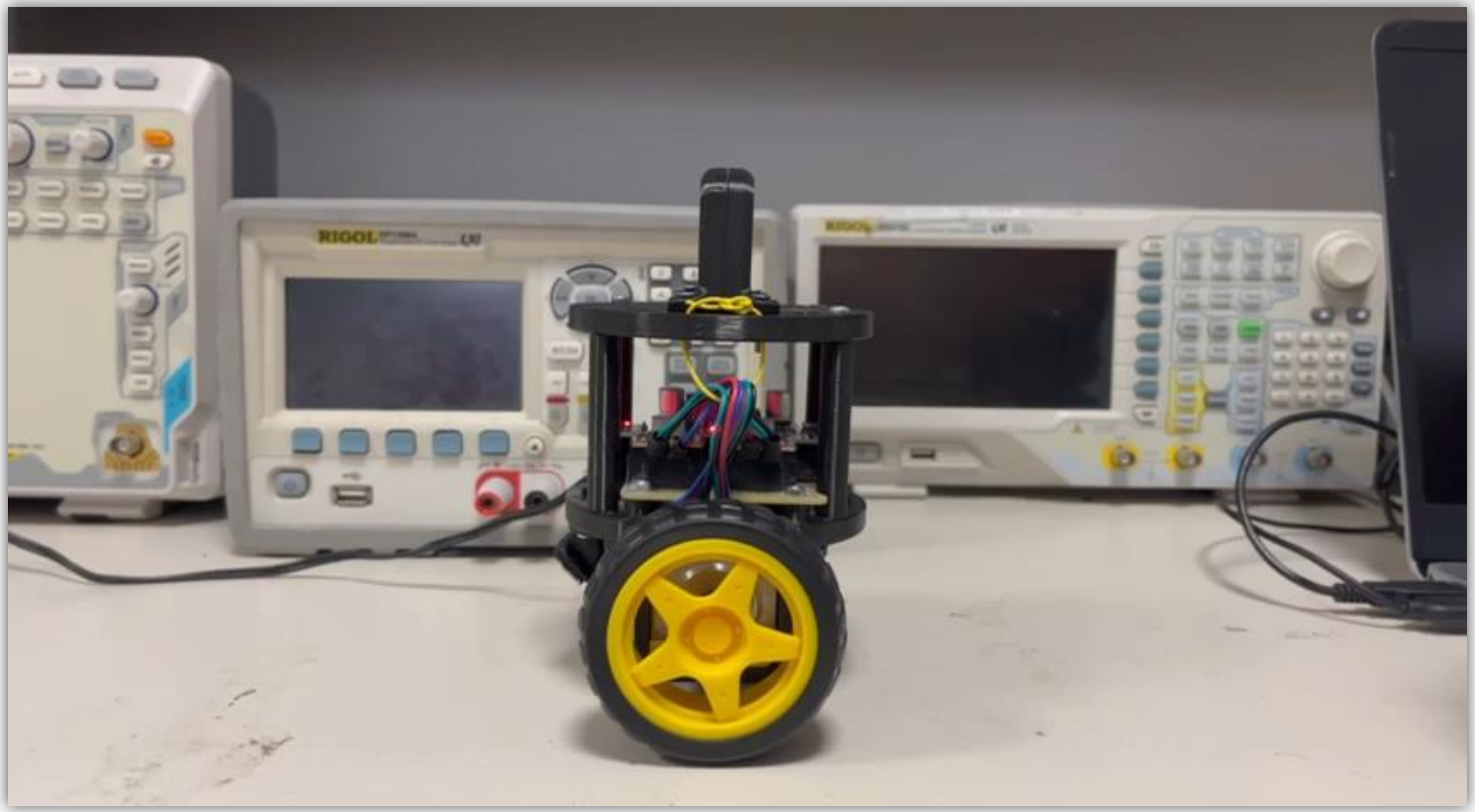
Sensor Data Acquisition

Low-pass filter + Kalman filter

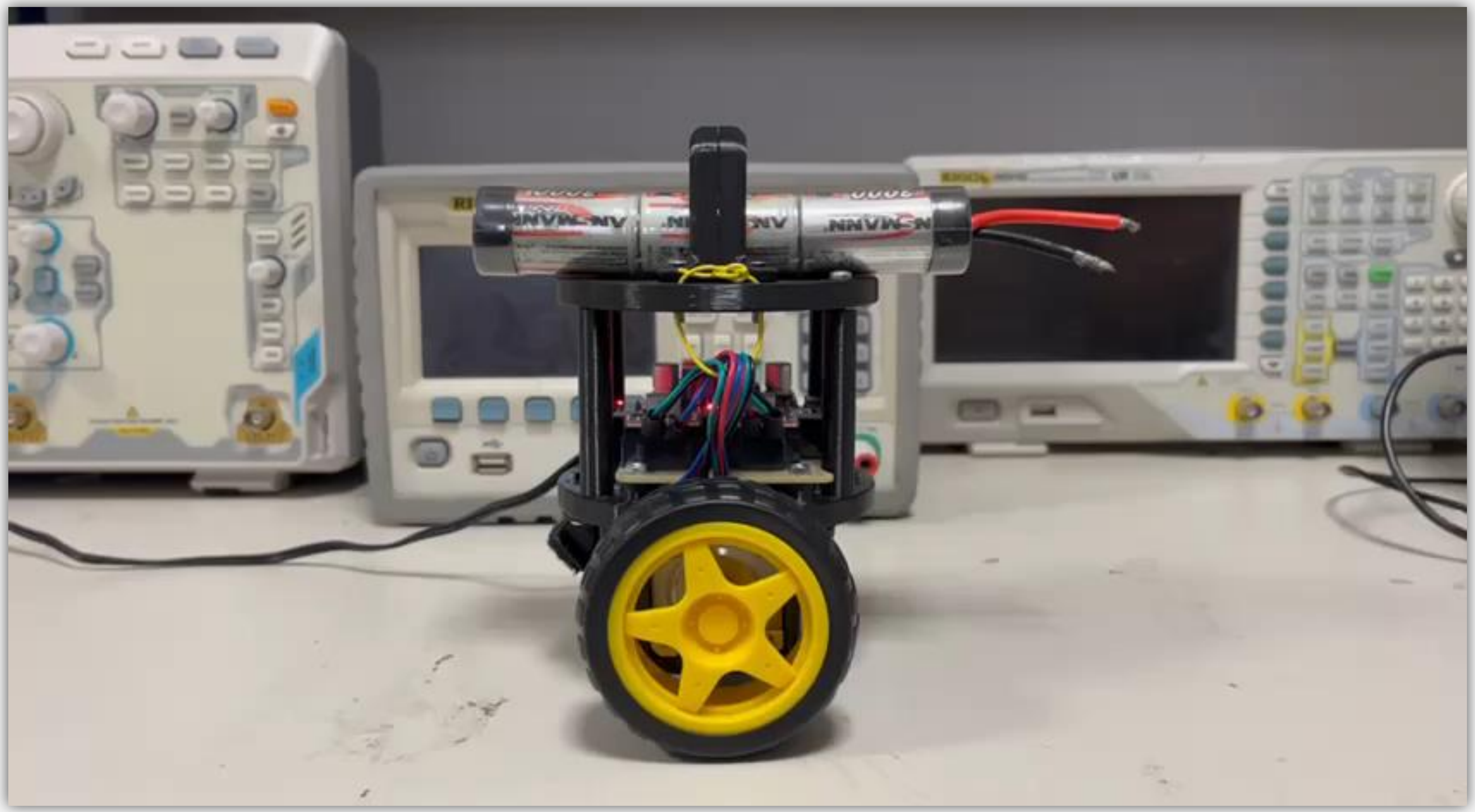
- Low-pass filter for high-frequency noise reduction (e.g. vibrations)
- Kalman filter for sensor data fusion and state reconstruction

Demo

SAPIENZA
UNIVERSITÀ DI ROMA



Demo

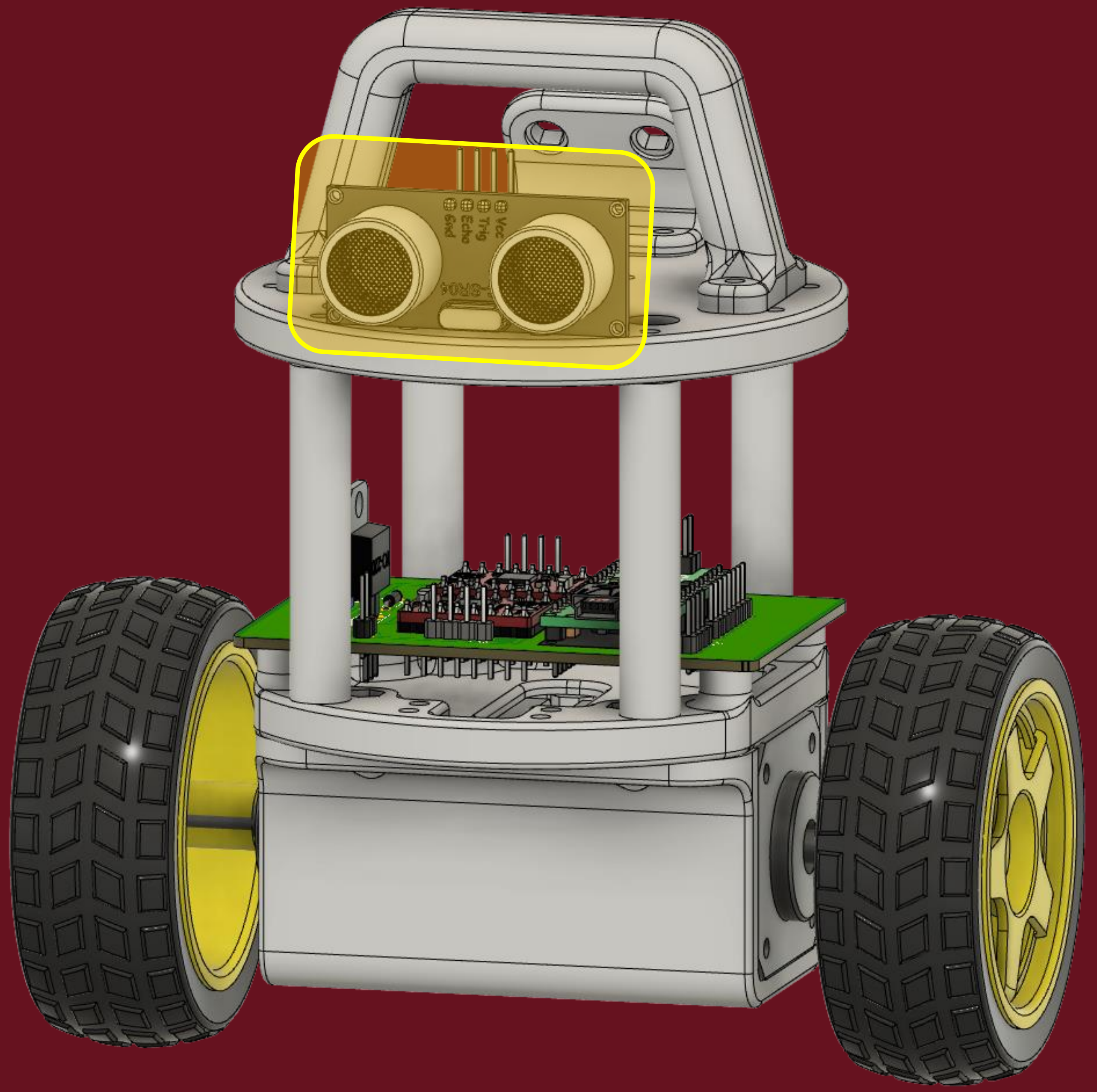


Future Developments

- Integrate an additional controller in cascade control configuration so that the robot can track a speed reference.

- Integrate an ultrasonic distance sensor (e.g. HC-SR04) to introduce the perception of its surrounding.

- Implement an Adaptive Cruise Control (ACC) algorithm to adjust the speed to maintain a safety distance from a moving target.



Conclusion

SAPIENZA
UNIVERSITÀ DI ROMA

The main objective of the project was to implement a control system to maintain balance for a wheeled robot.
We adopted the model-based design approach, using MATLAB and Simulink to simulate the controller's behaviour.
Our model proved to be accurate, reflecting the dynamics of the physical robot (some fine-tuning was necessary anyway).

During the implementation in Arduino, we faced significant challenges, particularly related to the limited computational capacity of the board.
However, through code optimization and streamlining, we successfully overcame these obstacles and got a working system.

The result is a robot that maintains balance in a robust fashion, demonstrated both in simulation and in the real world, proving that the model and the software are reliable, ensuring stability even in the face of external disturbance.

SAPIENZA
UNIVERSITÀ DI ROMA

Thanks For Your Attention