# RosMaster

CONEXIÓN
Conectar USB a Laptop y a Placa
Interruptor OFF
Conectar Bateria 12V
Interruptor ON

Para APAGAR
Interruptor OFF
Desconectar Bateria 12V
Desconectar USB

TERMINAL 1

source /opt/ros/humble/setup.bash

source ~/RosMaster/driver_ws/install/setup.bash

ros2 run yahboomcar_bringup Mcnamu_driver_X3

TERMINAL 2

source /opt/ros/humble/setup.bash

source ~/RosMaster/driver_ws/install/setup.bash

ros2 run rosmaster gyro

La carpeta llamada rosmaster esta dentro del src
La principal se llama RosMaster

Si se hacen modificaciones
SIEMPRE HACER COLCON BUILD
pero antes ELIMINAR las carpetas que hace el colcon build
Hay 2 Códigos
Con Yaw de 180 y Yaw de 90

Yaw 180 - gyro

Yaw 90 - gyro_cal

MEJOR CODIGO gyro_node.py

```
#!/usr/bin/env python3

import rclpy
from rclpy.node import Node
from sensor_msgs.msg import Imu, MagneticField
import math


class AttitudeMinimal(Node):

   def __init__(self):
       super().__init__('attitude_minimal')

       # Subscripciones
       self.create_subscription(Imu, '/imu/data_raw', self.imu_cb, 10)
       self.create_subscription(MagneticField, '/imu/mag', self.mag_cb,
10)

       # Estado
       self.roll = 0.0
       self.pitch = 0.0
       self.yaw = 0.0

       # Filtro (roll un poco más suave)
       self.alpha_rp = 0.05
       self.alpha_yaw = 0.1

       # Yaw referencia inicial
       self.yaw_offset = 0.0
       self.yaw_ref_set = False

       self.get_logger().info(
```

```
            'Nodo iniciado.\n'
            'Deja el IMU quieto mirando al frente durante 2 segundos.'
        )

    # ---------------- ACCEL ----------------
    def imu_cb(self, msg):
        ax = msg.linear_acceleration.x
        ay = msg.linear_acceleration.y
        az = msg.linear_acceleration.z

        # ---------- PITCH (NO TOCAR) ----------
        pitch = math.atan2(-ax, math.sqrt(ay**2 + az**2))

        # ---------- ROLL (invertido y estable) ----------
        roll = -math.atan2(ay, az)

        # Filtro exponencial
        self.pitch = self.alpha_rp * pitch + (1 - self.alpha_rp) *
self.pitch
        self.roll  = self.alpha_rp * roll  + (1 - self.alpha_rp) *
self.roll

        self.print_angles()

    # ---------------- MAG ----------------
    def mag_cb(self, msg):
        mx = msg.magnetic_field.x
        my = msg.magnetic_field.y

        # ---------- YAW (invertido) ----------
        yaw_raw = -math.atan2(my, mx)

        # Fijar yaw = 0 al frente inicial
        if not self.yaw_ref_set:
            self.yaw_offset = yaw_raw
            self.yaw_ref_set = True
            self.get_logger().info('Yaw cero fijado al frente inicial')
            return

        yaw = yaw_raw - self.yaw_offset

        # Normalizar a [-pi, pi]
        yaw = math.atan2(math.sin(yaw), math.cos(yaw))
```

```
        self.yaw = self.alpha_yaw * yaw + (1 - self.alpha_yaw) *
self.yaw

    # ---------------- PRINT ----------------
    def print_angles(self):
        self.get_logger().info(
            f'Roll:{math.degrees(self.roll):+6.1f}°  '
            f'Pitch:{math.degrees(self.pitch):+6.1f}°  '
            f'Yaw:{math.degrees(self.yaw):+6.1f}°'
        )


def main(args=None):
    rclpy.init(args=args)
    node = AttitudeMinimal()
    try:
        rclpy.spin(node)
    except KeyboardInterrupt:
        pass
    finally:
        node.destroy_node()
        rclpy.shutdown()


if __name__ == '__main__':
    main()
```