💻 CÓDIGO Y ALGORITMOS DE CONTROL MECATRÓNICO — MÓDULO 03

Curso: Curso Completo: Introducción a la Mecatrónica

Módulo 03: Módulo 03: Sensores Mecatrónicos: Finales de Carrera, Encoders, Ópticos e IMU

Archivos en `codigo_fuente/`:

- `01_firmware_arduino_modulo_03.ino` (Control en tiempo real sobre Arduino UNO)
- `02_supervision_mecatronica_modulo_03.py` (Supervisión HMI y registro CSV/SQLite en Raspberry Pi)
- `03_pruebas_y_calibracion_modulo_03.sh` (Script automatizado de validación de banco)

---

1. Arquitectura de Control y Flujo de Señales (2200×1350)



Plano 2 — Arquitectura de Control y Cinemática del Módulo 03

---

2. Firmware Completo de Tiempo Real (`01_firmware_arduino_modulo_03.ino`)

```
/*
 * ============================================================================
 * CURSO COMPLETO: INTRODUCCIÓN A LA MECATRÓNICA — MÓDULO 03
 * Archivo: 01_firmware_arduino_modulo_03.ino
 * Prototipo: Estación de Sensores Mecatrónicos (Homing, Encoder A/B, IR e IMU)
 * Hardware: Arduino UNO + Encoder (Pines 2,5) + Final de Carrera (Pin 3)
 *           + Sensor IR de Piezas (Pin 4) + IMU MPU6050 I2C (A4,A5)
 * ============================================================================
 */

#include <Arduino.h>
#include <Wire.h>

const uint8_t PIN_ENC_A         = 2; // INT0: Canal A del Encoder
const uint8_t PIN_FINAL_CARRERA = 3; // INT1: Microswitch de Homing (X = 0)
const uint8_t PIN_SENSOR_PIEZA  = 4; // Detector óptico de paso de piezas
const uint8_t PIN_ENC_B         = 5; // Canal B del Encoder (Sentido de giro)
const uint8_t PIN_ALARMA        = 8; // Indicador de tope o inclinación excesiva

const uint8_t MPU6050_ADDR      = 0x68;

volatile long posicionPasos            = 0;
volatile bool origenReferenciadoHoming = false;
volatile unsigned long ultimoHomingMs  = 0;

unsigned long contadorPiezas = 0;
bool estadoAnteriorPieza     = HIGH;
bool imuConectada            = false;
unsigned long ultimoEnvioMs  = 0;

// ISR 0: Conteo bidireccional en cuadratura
void isrEncoderCuadratura() {
  if (digitalRead(PIN_ENC_B) == HIGH) {
    posicionPasos++;
  } else {
    posicionPasos--;
  }
}

// ISR 1: Tope físico de referencia cero (Homing con antirrebote de 50 ms)
void isrFinalDeCarrera() {
  unsigned long ahora = millis();
  if (ahora - ultimoHomingMs > 50) {
    posicionPasos = 0;
    origenReferenciadoHoming = true;
    ultimoHomingMs = ahora;
  }
}

void inicializarMPU6050() {
  Wire.begin();
  Wire.beginTransmission(MPU6050_ADDR);
  Wire.write(0x6B); // Registro PWR_MGMT_1
  Wire.write(0x00); // Despertar al sensor MPU6050
  imuConectada = (Wire.endTransmission() == 0);
}
```

```
float leerInclinacionRollGrados() {
  if (!imuConectada) return 0.0;
  Wire.beginTransmission(MPU6050_ADDR);
  Wire.write(0x3B); // Registro ACCEL_XOUT_H
  if (Wire.endTransmission(false) != 0) return 0.0;

  Wire.requestFrom(MPU6050_ADDR, (uint8_t)6, (uint8_t)true);
  if (Wire.available() < 6) return 0.0;

  int16_t ax = (Wire.read() << 8) | Wire.read();
  int16_t ay = (Wire.read() << 8) | Wire.read();
  int16_t az = (Wire.read() << 8) | Wire.read();
  (void)ax;

  float rollRad = atan2((float)ay, (float)az);
  return rollRad * (180.0 / PI);
}

void setup() {
  Serial.begin(115200);

  pinMode(PIN_ENC_A, INPUT_PULLUP);
  pinMode(PIN_ENC_B, INPUT_PULLUP);
  pinMode(PIN_FINAL_CARRERA, INPUT_PULLUP);
  pinMode(PIN_SENSOR_PIEZA, INPUT_PULLUP);
  pinMode(PIN_ALARMA, OUTPUT);

  attachInterrupt(digitalPinToInterrupt(PIN_ENC_A), isrEncoderCuadratura, RISING);
  attachInterrupt(digitalPinToInterrupt(PIN_FINAL_CARRERA), isrFinalDeCarrera, FALLING);

  inicializarMPU6050();
  Serial.println(F("{"modulo":3,"evento":"SENSORES_MECATRONICOS_LISTOS"}"));
}

void loop() {
  // 1. Detección de piezas por flanco de bajada (no bloqueante)
  bool lecturaPieza = digitalRead(PIN_SENSOR_PIEZA);
  if (estadoAnteriorPieza == HIGH && lecturaPieza == LOW) {
    contadorPiezas++;
  }
  estadoAnteriorPieza = lecturaPieza;

  // 2. Envío periódico de telemetría unificada cada 200 ms
  unsigned long ahora = millis();
  if (ahora - ultimoEnvioMs >= 200) {
    ultimoEnvioMs = ahora;

    noInterrupts();
    long posActual = posicionPasos;
    bool homingOk  = origenReferenciadoHoming;
    interrupts();

    bool topeActivo = (digitalRead(PIN_FINAL_CARRERA) == LOW);
    float anguloImu = leerInclinacionRollGrados();
    bool alertaInclinacion = (fabs(anguloImu) > 25.0);

    digitalWrite(PIN_ALARMA, (topeActivo || alertaInclinacion) ? HIGH : LOW);
```

```
    Serial.print(F("{"modulo":3,"pos_pasos":"));
    Serial.print(posActual);
    Serial.print(F(","homing_ok":"));
    Serial.print(homingOk ? 1 : 0);
    Serial.print(F(","tope_activo":"));
    Serial.print(topeActivo ? 1 : 0);
    Serial.print(F(","piezas":"));
    Serial.print(contadorPiezas);
    Serial.print(F(","imu_deg":"));
    Serial.print(anguloImu, 1);
    Serial.println(F("}"));
  }
}
```

---

3. Software de Supervisión y Telemetría (`02_supervision_mecatronica_modulo_03.py`)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
================================================================================
CURSO COMPLETO: INTRODUCCIÓN A LA MECATRÓNICA — MÓDULO 03
Archivo: 02_supervision_mecatronica_modulo_03.py
Descripción: Monitor de instrumentación mecatrónica en Raspberry Pi.
             Supervisa Homing, coordenadas de encoder en cuadratura,
             conteo de piezas e inclinación del chasis (MPU6050).
================================================================================
"""

import csv
import json
import os
import sys
import time
from datetime import datetime

try:
    import serial
except ImportError:
    serial = None

PUERTO_SERIE = os.environ.get("MECATRONICA_PORT", "/dev/ttyACM0")
BAUDIOS      = 115200
ARCHIVO_CSV  = "instrumentacion_modulo_03.csv"


def inicializar_csv(ruta: str) -> None:
    if not os.path.exists(ruta):
        with open(ruta, mode="w", newline="", encoding="utf-8") as f:
            csv.writer(f).writerow(["timestamp", "pos_pasos", "homing_ok", "tope_activo", "piezas",
"imu_deg"])


def registrar_lectura(d: dict) -> None:
    ts   = datetime.now().strftime("%H:%M:%S.%f")[:-3]
    pos  = d.get("pos_pasos", 0)
    hom  = d.get("homing_ok", 0)
    tope = d.get("tope_activo", 0)
    pzs  = d.get("piezas", 0)
    imu  = d.get("imu_deg", 0.0)

    alerta = "⚠️ INCLINACIÓN > 25°" if abs(imu) > 25.0 else ("🛑 TOPE PULSADO (X=0)" if tope else
"OK")
    print(
        f"[{ts}] Encoder:{pos:+5d} pasos | Homing:{'SÍ' if hom else 'NO'} | "
        f"Piezas:{pzs:3d} | IMU:{imu:+5.1f}° | {alerta}"
    )
    with open(ARCHIVO_CSV, mode="a", newline="", encoding="utf-8") as f:
        csv.writer(f).writerow([ts, pos, hom, tope, pzs, imu])


def ejecutar_demo() -> None:
    print("ℹ️ Simulando secuencia de sensores mecatrónicos (Encoder + Homing + IR + MPU6050)...")
    secuencia = [
```

```
        {"pos_pasos": -42, "homing_ok": 0, "tope_activo": 0, "piezas": 0, "imu_deg": 1.2},
        {"pos_pasos": 0,   "homing_ok": 1, "tope_activo": 1, "piezas": 1, "imu_deg": 1.4},
        {"pos_pasos": 120, "homing_ok": 1, "tope_activo": 0, "piezas": 2, "imu_deg": 3.5},
        {"pos_pasos": 200, "homing_ok": 1, "tope_activo": 0, "piezas": 3, "imu_deg": 28.4},
    ]
    for s in secuencia:
        registrar_lectura(s)
        time.sleep(0.35)


def main() -> None:
    print("=" * 78)
    print(" MONITOR DE SENSORES MECATRÓNICOS MÓDULO 03 — HOMING, ENCODER E IMU")
    print("=" * 78)
    inicializar_csv(ARCHIVO_CSV)
    if "--demo" in sys.argv or serial is None:
        ejecutar_demo()
        return
    try:
        with serial.Serial(PUERTO_SERIE, BAUDIOS, timeout=1.0) as ser:
            while True:
                linea = ser.readline().decode("utf-8", errors="ignore").strip()
                if linea.startswith("{") and "pos_pasos" in linea:
                    registrar_lectura(json.loads(linea))
    except Exception as e:
        print(f"⚠️ Sin enlace serie físico ({e}). Ejecutando simulación:")
        ejecutar_demo()


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