💻 CÓDIGO Y ALGORITMOS DE CONTROL MECATRÓNICO — MÓDULO 01

Curso: Curso Completo: Introducción a la Mecatrónica

Módulo 01: Módulo 01: ¿Qué es la Mecatrónica? Los 4 Pilares, Lazo Cerrado y Banco de Pruebas

Archivos en `codigo_fuente/`:

- `01_firmware_arduino_modulo_01.ino` (Control en tiempo real sobre Arduino UNO)
- `02_supervision_mecatronica_modulo_01.py` (Supervisión HMI y registro CSV/SQLite en Raspberry Pi)
- `03_pruebas_y_calibracion_modulo_01.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 01

---

2. Firmware Completo de Tiempo Real (`01_firmware_arduino_modulo_01.ino`)

```
/*
 * ============================================================================
 * CURSO COMPLETO: INTRODUCCIÓN A LA MECATRÓNICA — MÓDULO 01
 * Archivo: 01_firmware_arduino_modulo_01.ino
 * Prototipo: Maqueta de Sistema Mecatrónico en Lazo Cerrado (Los 4 Pilares)
 * Hardware: Arduino UNO R3 + Servomotor (Pin 9) + Potenciómetro (A0)
 *           + Sensor Óptico de Confirmación (Pin 4) + E-Stop (Pin 2)
 * ============================================================================
 */

#include <Arduino.h>
#include <Servo.h>

// --- Definición de Pines Físicos ---
const uint8_t PIN_POT_SETPOINT = A0;  // Consigna deseada (0..1023)
const uint8_t PIN_ESTOP        = 2;   // Botón Parada de Emergencia (INPUT_PULLUP)
const uint8_t PIN_SENSOR_META  = 4;   // Sensor óptico detector de llegada a 90°
const uint8_t PIN_LED_VERDE    = 8;   // Indicador de operación normal
const uint8_t PIN_SERVO_PWM    = 9;   // Actuador servomotor (Brazo 1 DoF)
const uint8_t PIN_LED_ROJO     = 10;  // Indicador de Parada de Emergencia / Falla

Servo brazoMecatronico;

// --- Variables de Estado del Lazo Cerrado ---
int anguloSetpoint      = 0;
int anguloActual        = 0;
bool enPosicionMeta     = false;
bool paradaEmergencia   = false;
int comandoRemotoGrados = -1; // -1 = Usar potenciómetro local

unsigned long ultimoEnvioMs = 0;
const unsigned long INTERVALO_TELEMETRIA_MS = 200;

void procesarComandosSerie() {
  if (Serial.available() > 0) {
    String cmd = Serial.readStringUntil('\n');
    cmd.trim();
    cmd.toUpperCase();
    if (cmd == "AUTO") {
      comandoRemotoGrados = -1;
    } else if (cmd == "STOP") {
      paradaEmergencia = true;
    } else if (cmd == "RESET") {
      paradaEmergencia = false;
    } else if (cmd.startsWith("SET:")) {
      int val = cmd.substring(4).toInt();
      comandoRemotoGrados = constrain(val, 0, 180);
    }
  }
}

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

  pinMode(PIN_ESTOP, INPUT_PULLUP);
  pinMode(PIN_SENSOR_META, INPUT_PULLUP);
  pinMode(PIN_LED_VERDE, OUTPUT);
```

```
  pinMode(PIN_LED_ROJO, OUTPUT);

  brazoMecatronico.attach(PIN_SERVO_PWM);
  brazoMecatronico.write(0); // Posición inicial segura (0 grados)

  digitalWrite(PIN_LED_VERDE, HIGH);
  digitalWrite(PIN_LED_ROJO, LOW);

  Serial.println(F("{"modulo":1,"evento":"INICIO_SISTEMA_LAZO_CERRADO","estado":"OK"}"));
}

void loop() {
  procesarComandosSerie();

  // 1. PILAR DE SEGURIDAD: Verificar botón físico de Parada de Emergencia
  if (digitalRead(PIN_ESTOP) == LOW) {
    paradaEmergencia = true;
  }

  if (paradaEmergencia) {
    // Llevar el mecanismo a posición segura de reposo (0°)
    brazoMecatronico.write(0);
    anguloActual = 0;
    digitalWrite(PIN_LED_VERDE, LOW);
    digitalWrite(PIN_LED_ROJO, HIGH);
  } else {
    // 2. PILAR DE CONTROL: Leer Setpoint (local por potenciómetro o remoto por USB)
    if (comandoRemotoGrados >= 0) {
      anguloSetpoint = comandoRemotoGrados;
    } else {
      int lecturaAdc = analogRead(PIN_POT_SETPOINT);
      anguloSetpoint = map(lecturaAdc, 0, 1023, 0, 180);
    }

    // 3. PILAR DE ACTUACIÓN: Mover eslabón mecánico con rampa suave
    if (anguloActual < anguloSetpoint) {
      anguloActual = min(anguloActual + 3, anguloSetpoint);
    } else if (anguloActual > anguloSetpoint) {
      anguloActual = max(anguloActual - 3, anguloSetpoint);
    }
    brazoMecatronico.write(anguloActual);

    // 4. PILAR DE PERCEPCIÓN: Leer sensor óptico de confirmación física
    enPosicionMeta = (digitalRead(PIN_SENSOR_META) == LOW);

    digitalWrite(PIN_LED_VERDE, HIGH);
    digitalWrite(PIN_LED_ROJO, enPosicionMeta ? LOW : LOW);
  }

  // 5. PILAR DE COMPUTACIÓN: Enviar telemetría JSON no bloqueante a Raspberry Pi
  unsigned long ahora = millis();
  if (ahora - ultimoEnvioMs >= INTERVALO_TELEMETRIA_MS) {
    ultimoEnvioMs = ahora;
    int errorGrados = anguloSetpoint - anguloActual;
    Serial.print(F("{"modulo":1,"setpoint_deg":"));
    Serial.print(anguloSetpoint);
    Serial.print(F(","actual_deg":"));
    Serial.print(anguloActual);
```

```
    Serial.print(F(","error_deg":"));
    Serial.print(errorGrados);
    Serial.print(F(","meta_ok":"));
    Serial.print(enPosicionMeta ? 1 : 0);
    Serial.print(F(","estop":"));
    Serial.print(paradaEmergencia ? 1 : 0);
    Serial.println(F("}"));
  }

  delay(15); // Paso de actualización suave del servomotor
}
```

---

3. Software de Supervisión y Telemetría (`02_supervision_mecatronica_modulo_01.py`)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
================================================================================
CURSO COMPLETO: INTRODUCCIÓN A LA MECATRÓNICA — MÓDULO 01
Archivo: 02_supervision_mecatronica_modulo_01.py
Descripción: Supervisor HMI en Raspberry Pi para la maqueta mecatrónica en
             lazo cerrado. Recibe tramas JSON por USB, registra historial CSV
             y calcula la precisión del posicionamiento angular.
================================================================================
"""

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  = "registro_mecatronica_modulo_01.csv"


def inicializar_csv(ruta: str) -> None:
    if not os.path.exists(ruta):
        with open(ruta, mode="w", newline="", encoding="utf-8") as f:
            writer = csv.writer(f)
            writer.writerow(["timestamp", "setpoint_deg", "actual_deg", "error_deg", "meta_ok",
"estop"])


def procesar_trama(datos: dict) -> None:
    ts = datetime.now().strftime("%Y-%m-%d %H:%M:%S.%f")[:-3]
    sp = datos.get("setpoint_deg", 0)
    ac = datos.get("actual_deg", 0)
    er = datos.get("error_deg", 0)
    ok = datos.get("meta_ok", 0)
    es = datos.get("estop", 0)

    estado_str = "🚨 PARADA EMERGENCIA" if es else ("✅ EN META (90°)" if ok else "🔄 EN
MOVIMIENTO")
    print(f"[{ts}] Setpoint: {sp:3d}° | Actual: {ac:3d}° | Error: {er:+4d}° | Estado: {estado_str}")

    with open(ARCHIVO_CSV, mode="a", newline="", encoding="utf-8") as f:
        writer = csv.writer(f)
        writer.writerow([ts, sp, ac, er, ok, es])


def modo_simulacion_didactica() -> None:
    print("ℹ️ Ejecutando simulación cinemática de Lazo Cerrado (Módulo 01)...")
    muestras = [
```

```
        {"setpoint_deg": 90, "actual_deg": 30, "error_deg": 60, "meta_ok": 0, "estop": 0},
        {"setpoint_deg": 90, "actual_deg": 65, "error_deg": 25, "meta_ok": 0, "estop": 0},
        {"setpoint_deg": 90, "actual_deg": 90, "error_deg": 0,  "meta_ok": 1, "estop": 0},
        {"setpoint_deg": 90, "actual_deg": 0,  "error_deg": 90, "meta_ok": 0, "estop": 1},
    ]
    for m in muestras:
        procesar_trama(m)
        time.sleep(0.4)


def main() -> None:
    print("=" * 74)
    print(" SUPERVISOR MECATRÓNICO MÓDULO 01 — LOS 4 PILARES Y LAZO CERRADO")
    print("=" * 74)
    inicializar_csv(ARCHIVO_CSV)

    if "--demo" in sys.argv or serial is None:
        modo_simulacion_didactica()
        return

    try:
        with serial.Serial(PUERTO_SERIE, BAUDIOS, timeout=1.0) as ser:
            print(f"✅ Conectado a {PUERTO_SERIE} ({BAUDIOS} bps). Presiona Ctrl+C para salir.")
            while True:
                linea = ser.readline().decode("utf-8", errors="ignore").strip()
                if linea.startswith("{") and linea.endswith("}"):
                    try:
                        datos = json.loads(linea)
                        if "setpoint_deg" in datos:
                            procesar_trama(datos)
                    except json.JSONDecodeError:
                        pass
    except Exception as exc:
        print(f"⚠️ Puerto serie {PUERTO_SERIE} no disponible ({exc}). Activando modo
demostración:")
        modo_simulacion_didactica()


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