

Còdigo

// Control motor DC L298N: modo normal (potencia segun pot) y modo asistencia con inversion + rampa

const int pinPot = A0;

const int pinBtn = 2;      // boton (con INPUT\_PULLUP): cuando LOW = presionado

const int pinIN1 = 8;

const int pinIN2 = 7;

const int pinENA = 9;      // PWM

// Rampa

int appliedPWM = 0;

const int rampStep = 5;         // incremento por paso (ajustable)

const int rampDelayMs = 30;     // tiempo entre pasos (ajustable)

// límites de seguridad

const int PWM\_MIN = 0;

const int PWM\_MAX = 230;       // puedes limitar <255 para proteger motor

void setup() {

  pinMode(pinIN1, OUTPUT);

  pinMode(pinIN2, OUTPUT);

  pinMode(pinENA, OUTPUT);

  pinMode(pinBtn, INPUT\_PULLUP); // usa resistencia interna, boton a GND

  analogWrite(pinENA, 0);

  digitalWrite(pinIN1, LOW);

  digitalWrite(pinIN2, LOW);

  Serial.begin(115200);

  delay(100);

  Serial.println("time\_ms,pot\_voltage,rawPot,targetPWM,appliedPWM,mode,direction");

}

void loop() {

  unsigned long t = millis();

  int rawPot = analogRead(pinPot);            // 0..1023

  float potV = rawPot \* (5.0 / 1023.0);       // aproximado

  int targetPWM = map(rawPot, 0, 1023, PWM\_MIN, PWM\_MAX);

  bool btnPressed = (digitalRead(pinBtn) == LOW); // true si presionado

  if (btnPressed) {

    // MODO ASISTENCIA: invertir sentido y rampa hacia targetPWM

    setDirectionReverse();

    // rampa hacia target

    rampTo(targetPWM);

    // modo string

    logSerial(t, potV, rawPot, targetPWM, appliedPWM, "ASIST", "REV");

  } else {

    // MODO NORMAL: sentido forward, velocidad igual pot (sin rampa o con rampa opcional)

    setDirectionForward();

    // Si quieres, usa rampa para suavizar transiciones:

    rampTo(targetPWM);

    logSerial(t, potV, rawPot, targetPWM, appliedPWM, "NORMAL", "FWD");

  }

  delay(20); // loop breve

}

// Funciones auxiliares

void setDirectionForward() {

  digitalWrite(pinIN1, HIGH);

  digitalWrite(pinIN2, LOW);

}

void setDirectionReverse() {

  digitalWrite(pinIN1, LOW);

  digitalWrite(pinIN2, HIGH);

}

void rampTo(int target) {

  if (appliedPWM < target) {

    while (appliedPWM < target) {

      appliedPWM += rampStep;

      if (appliedPWM > target) appliedPWM = target;

      analogWrite(pinENA, appliedPWM);

      delay(rampDelayMs);

    }

  } else if (appliedPWM > target) {

    while (appliedPWM > target) {

      appliedPWM -= rampStep;

      if (appliedPWM < target) appliedPWM = target;

      analogWrite(pinENA, appliedPWM);

      delay(rampDelayMs);

    }

  } else {

    analogWrite(pinENA, appliedPWM);

  }

}

void logSerial(unsigned long t, float potV, int rawPot, int targetPWM, int appliedPWM, const char\* mode, const char\* dir) {

  // CSV: time\_ms,pot\_voltage,rawPot,targetPWM,appliedPWM,mode,direction

  Serial.print(t); Serial.print(",");

  Serial.print(potV,3); Serial.print(",");

  Serial.print(rawPot); Serial.print(",");

  Serial.print(targetPWM); Serial.print(",");

  Serial.print(appliedPWM); Serial.print(",");

  Serial.print(mode); Serial.print(",");

  Serial.println(dir);

}