long ticksMesures = ticks; *copie atomique ticks = 0;* RAZ pour prochaine fenêtre

float vitesse = ticksMesures * (1000.0 / period); *en ticks/s*

PID

erreur = consigne - vitesse; integral += erreur * (period / 1000.0); float deriv = (erreur - erreurPrec) / (period / 1000.0); erreurPrec = erreur;

float commande = kp * erreur + ki * integral + kd * deriv;

Limiter entre -255 et 255 if (commande > 255) commande = 255; if (commande < -255) commande = -255; commandeMoteur(commande); — Affichage TP — Serial.print(“Consigne=”); Serial.print(consigne); Serial.print(“ | Vitesse=”); Serial.print(vitesse); Serial.print(“ | PWM=”); Serial.println(commande);

Code Arduino du TP

```
 // === TP : PID pour régulation de VITESSE d’un moteur CC === // Mesure vitesse = codeur incrémental sur interruption // Consigne reçue par le Moniteur Série (en tr/min ou en ticks/s)

// --- Pont en H ---
const int M_AV = 3; // PWM forward
const int M_AR = 6; // PWM reverse

// --- Codeur incrémental ---
const int canalA = 2; // interruption 0
const int canalB = 11;

volatile long ticks = 0; // compteur modifié par ISR

// === PID ===
float consigne = 0; // vitesse ciblée (ex : en ticks/s)
float kp = 0.8; // gains PID : à régler en TP
float ki = 0.1;
float kd = 0.05;

float erreur, erreurPrec = 0;
float integral = 0;

// === Mesure période ===
unsigned long lastMeasure = 0;
const unsigned long period = 100; // calcul vitesse toutes les 100 ms

// === Prototypes ===
void ISR_codeur();
void commandeMoteur(float pwm);
```

```
float lireConsigne();

void setup() {
Serial.begin(9600);

pinMode(M_AV, OUTPUT);
pinMode(M_AR, OUTPUT);

pinMode(canalB, INPUT);

attachInterrupt(digitalPinToInterrupt(canalA), ISR_codeur, RISING);

Serial.println("=== TP : Regulation PID de vitesse ===");
Serial.println("Entrez une consigne en ticks/s (ex : 200):");
}

// ====================== BOUCLE PRINCIPALE ==========================
void loop() {
// --- Lecture consigne si disponible ---
if (Serial.available() > 0) {
consigne = lireConsigne();
Serial.print("Nouvelle consigne = ");
Serial.println(consigne);
}

// --- Boucle PID toutes les 100 ms ---
unsigned long now = millis();
if (now - lastMeasure >= period) {
lastMeasure = now;

long ticksMesures = ticks; // copie atomique
ticks = 0;                 // RAZ pour prochaine fenêtre

float vitesse = ticksMesures * (1000.0 / period);  // en ticks/s

// ================ PID ================
erreur = consigne - vitesse;
integral += erreur * (period / 1000.0);
float deriv = (erreur - erreurPrec) / (period / 1000.0);
erreurPrec = erreur;

float commande = kp * erreur + ki * integral + kd * deriv;

// Limiter entre -255 et 255
if (commande > 255) commande = 255;
if (commande < -255) commande = -255;

commandeMoteur(commande);
```

```
// --- Affichage TP ---
Serial.print("Consigne=");
Serial.print(consigne);
Serial.print("  | Vitesse=");
Serial.print(vitesse);
Serial.print("  | PWM=");
Serial.println(commande);


}
}

// ========== INTERRUPTIONS CODEUR ==========
void ISR_codeur() {
if (digitalRead(canalB))
ticks++;
else
ticks--;
}

// ========== COMMANDE MOTEUR ==========
void commandeMoteur(float pwm) {
if (pwm >= 0) {
digitalWrite(M_AR, LOW);
analogWrite(M_AV, pwm);
} else {
digitalWrite(M_AV, LOW);
analogWrite(M_AR, -pwm);
}
}

// ========== LECTURE CONSIGNE ==========
float lireConsigne() {
String txt = Serial.readStringUntil('\n');
txt.trim();
return txt.toFloat();
}
```