```
#include "mCoreLite.h"

mCoreLite bot;

// ----- CONSIGNE -----
float Dcons = 20.0;      // distance cible (cm)

// ----- PARAMÈTRES PID -----
float Kp = 6.0;          // proportionnel
float Ki = 0.15;         // intégral
float Kd = 0;          // dérivé (amortissement)

// ----- MEMOIRES PID -----
float integral = 0.0;
float e_prev = 0.0;
unsigned long lastTime = 0;

// ----- SECURITE INTEGRALE (ANTI-WINDUP) -----
float Imax = 50.0;       // limite de l'intégrale

// ----- VITESSE -----
int Vmin = 60;           // évite les sifflements
int Vmax = 200;          // saturation

// ----- FILTRE DERIVÉ -----
// (évite les tremblements dus au bruit US)
float alpha = 0.7;       // 0 = pas de filtre, 1 = filtre fort
float d_filtered = 0.0;

void setup() {
  Serial.begin(115200);
  bot.botSetup();
  lastTime = millis();
}

void loop() {

  // ----- 1) MESURE -----
  float Dmes = bot.distanceCM();

  // ----- 2) ERREUR -----
  float e = Dmes - Dcons;

  // ----- 3) TEMPS -----
  unsigned long now = millis();
  float dt = (now - lastTime) / 1000.0;
  if (dt <= 0) dt = 0.001;   // sécurité
  lastTime = now;

  // ----- 4) ZONE MORTE -----
  if (fabs(e) < 1.0) {
```

```
    integral = 0;           // détendre l'intégrale
    bot.motors(0,0);
    debug(Dmes, e, 0, integral, 0);
    return;
  }

  // ----- 5) INTÉGRALE + ANTI-WINDUP -----
  integral += e * dt;
  if (integral >  Imax) integral =  Imax;
  if (integral < -Imax) integral = -Imax;

  // Réinitialisation si inversion de signe de l'erreur
  if (e * e_prev < 0) integral = 0;

  // ----- 6) DÉRIVÉE (/!\ capteur bruyant) -----
  float derivative = (e - e_prev) / dt;
  e_prev = e;

  // Filtre passe-bas sur le dérivé
  d_filtered = alpha * d_filtered + (1 - alpha) * derivative;

  // ----- 7) SORTIE PID -----
  float v = Kp * e + Ki * integral + Kd * d_filtered;

  // ----- 8) SATURATION -----
  if (v >  Vmax) v =  Vmax;
  if (v < -Vmax) v = -Vmax;

  // ----- 9) VITESSE MINIMALE -----
  if (fabs(v) > 0 && fabs(v) < Vmin) {
    v = (v > 0) ? Vmin : -Vmin;
  }

  // ----- 10) ACTION -----
  // v > 0 → avancer (cible trop loin)
  // v < 0 → reculer (cible trop proche)
  bot.motors(-v, v);

  // ----- 11) DEBUG -----
  debug(Dmes, e, v, integral, d_filtered);
}

// ================== DEBUG ======================
void debug(float D, float e, float v, float I, float d) {
  Serial.print("D=");
  Serial.print(D);
  Serial.print("  e=");
  Serial.print(e);
  Serial.print("  v=");
```

```
  Serial.print(v);
  Serial.print("  I=");
  Serial.print(I);
  Serial.print("  d=");
  Serial.println(d);
}
```