```
// ======================================================
// CARRO BOMBEIRO DETECTOR DE CHAMA - Estático com alinhamento + aproximação
// - Evita ficar parado após girar
// - Gira em passos curtos, verifica e faz pequenos avanços ("nudges")
// ======================================================

#define FL_LEFT   A0
#define FL_CENTER A1
#define FL_RIGHT  A2

#define ENA 10
#define IN1 9
#define IN2 8
#define IN3 7
#define IN4 6
#define ENB 5

#define PUMP 4

// ---- Velocidades e tempos ----
int SPEED_TURN   = 120;   // velocidade de giro
int SPEED_FWD    = 170;   // velocidade de avanço
int THRESH_FIRE  = 400;   // limite de detecção (ajuste conforme calibração)
int ATTACK_TIME  = 2000;  // tempo que a bomba ficará ligada (ms)
int TURN_MS      = 250;   // duração de cada giro curto (antes: 350)
int ADVANCE_MS   = 250;   // tempo de avanço frontal curto (antes: 300)
int NUDGE_MS     = 180;   // micro impulso para destravar ou reposicionar

// ---- Estratégia de alinhamento e aproximação ----
int MAX_ALIGN_STEPS  = 8;   // tentativas máximas de alinhamento
int APPROACH_PULSES  = 4;   // avanços curtos antes de atacar
int DIR_MARGIN       = 20;  // margem para decidir esquerda/direita quando valores próximos
int CENTER_MARGIN    = 30;  // diferença mínima para considerar o sensor central "alinhado"

// ---- Variáveis globais ----
int leftVal, centerVal, rightVal;

void setup() {
```

```
  Serial.begin(9600);
  Serial.println(F("=== CARRO BOMBEIRO - Alinhamento + Aproximação
==="));

  pinMode(FL_LEFT,   INPUT);
  pinMode(FL_CENTER, INPUT);
  pinMode(FL_RIGHT,  INPUT);

  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);

  pinMode(PUMP, OUTPUT);
  digitalWrite(PUMP, LOW);

  StopMotores();
  delay(400);
}

void loop() {
  readFlames(leftVal, centerVal, rightVal);

  Serial.print("E:"); Serial.print(leftVal);
  Serial.print("\tC:"); Serial.print(centerVal);
  Serial.print("\tD:"); Serial.println(rightVal);

  bool fogoDetectado = (leftVal < THRESH_FIRE) || (centerVal <
THRESH_FIRE) || (rightVal < THRESH_FIRE);

  if (!fogoDetectado) {
    StopMotores();
    digitalWrite(PUMP, LOW);
    delay(80);
    return;
  }

  // 1) Alinhar-se em direção ao lado com maior detecção de fogo
  alignAndApproach();

  delay(80);
```

```
}

// ===================== Núcleo da estratégia =====================

void alignAndApproach() {
  // Faz nova leitura para decidir direção com média
  readFlames(leftVal, centerVal, rightVal);

  // Se já estiver centrado (sensor central claramente menor)
  if (isCentered(leftVal, centerVal, rightVal)) {
    Serial.println(F("🔥 Centro detectado: avanço curto + ataque"));
    adelante(SPEED_FWD); delay(ADVANCE_MS); StopMotores();
    attack();
    return;
  }

  // Se valores dos lados forem parecidos, faz pequeno avanço
  if (abs(leftVal - rightVal) <= DIR_MARGIN) {
    Serial.println(F("⚖️ Lados equilibrados: pequeno avanço"));
    adelante(SPEED_FWD); delay(NUDGE_MS); StopMotores();
    readFlames(leftVal, centerVal, rightVal);
    if (isCentered(leftVal, centerVal, rightVal)) {
      adelante(SPEED_FWD); delay(ADVANCE_MS); StopMotores();
      attack();
      return;
    }
  }

  // 2) Giro incremental com reavaliação (até MAX_ALIGN_STEPS)
  for (int i = 0; i < MAX_ALIGN_STEPS; i++) {
    readFlames(leftVal, centerVal, rightVal);

    // decisão de direção
    if ((rightVal + DIR_MARGIN) < leftVal) {
      Serial.println(F("↪️ Girando para DIREITA (passo)"));
      direita(SPEED_TURN); delay(TURN_MS); StopMotores();
    } else if ((leftVal + DIR_MARGIN) < rightVal) {
      Serial.println(F("↩️ Girando para ESQUERDA (passo)"));
      esquerda(SPEED_TURN); delay(TURN_MS); StopMotores();
    } else {
      // valores muito próximos → pequeno avanço
      Serial.println(F("⚖️ Equilíbrio durante alinhamento: pequeno
avanço"));
```

```
      adelante(SPEED_FWD); delay(NUDGE_MS); StopMotores();
    }

    // verifica se já centralizou
    readFlames(leftVal, centerVal, rightVal);
    if (isCentered(leftVal, centerVal, rightVal)) {
      Serial.println(F("✅ Alinhado após giro"));
      break;
    }
  }

  // 3) Aproximação em pulsos curtos com verificação contínua
  for (int k = 0; k < APPROACH_PULSES; k++) {
    readFlames(leftVal, centerVal, rightVal);

    if (!fireVisible(leftVal, centerVal, rightVal)) {
      Serial.println(F("❌ Fogo perdido na aproximação, parando"));
      break;
    }

    // correção de leve caso o centro não seja o menor
    if (!isCentered(leftVal, centerVal, rightVal)) {
      if (rightVal < leftVal - DIR_MARGIN) {
        direita(SPEED_TURN); delay(TURN_MS); StopMotores();
      } else if (leftVal < rightVal - DIR_MARGIN) {
        esquerda(SPEED_TURN); delay(TURN_MS); StopMotores();
      }
    }

    // pequeno avanço
    adelante(SPEED_FWD); delay(ADVANCE_MS); StopMotores();

    // se já está bem centralizado e leitura muito baixa → ataque
    readFlames(leftVal, centerVal, rightVal);
    if (isCentered(leftVal, centerVal, rightVal) && centerVal <
THRESH_FIRE) {
      Serial.println(F("🚀 Alvo central confirmado: iniciar ataque"));
      attack();
      return;
    }
  }

  // se não atacou, retorna ao modo de espera
```

```
  StopMotores();
}

// --- Funções auxiliares de decisão ---
bool isCentered(int L, int C, int R) {
  // Centro suficientemente menor que os lados
  return (C + CENTER_MARGIN < L) && (C + CENTER_MARGIN < R);
}

bool fireVisible(int L, int C, int R) {
  // Há chama visível em algum sensor
  return (L < THRESH_FIRE) || (C < THRESH_FIRE) || (R < THRESH_FIRE);
}

// ====================== Movimento ======================

void adelante(int vel) {
  analogWrite(ENA, vel);
  analogWrite(ENB, vel);
  // mapeamento invertido conforme ligação atual dos motores
  digitalWrite(IN1, LOW);  digitalWrite(IN2, HIGH);  // motor esquerdo
para frente
  digitalWrite(IN3, LOW);  digitalWrite(IN4, HIGH);  // motor direito
para frente
}

void atras(int vel) {
  analogWrite(ENA, vel);
  analogWrite(ENB, vel);
  digitalWrite(IN1, HIGH); digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH); digitalWrite(IN4, LOW);
}

void esquerda(int vel) {
  analogWrite(ENA, vel);
  analogWrite(ENB, vel);
  digitalWrite(IN1, HIGH); digitalWrite(IN2, LOW);   // esquerdo para
trás
  digitalWrite(IN3, LOW);  digitalWrite(IN4, HIGH);  // direito para
frente
}

void direita(int vel) {
```

```
  analogWrite(ENA, vel);
  analogWrite(ENB, vel);
  digitalWrite(IN1, LOW);  digitalWrite(IN2, HIGH);  // esquerdo para
frente
  digitalWrite(IN3, HIGH); digitalWrite(IN4, LOW);   // direito para
trás
}

void StopMotores() {
  analogWrite(ENA, 0);
  analogWrite(ENB, 0);
  digitalWrite(IN1, LOW); digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW); digitalWrite(IN4, LOW);
}

// ===================== Ação da bomba =====================

void attack() {
  Serial.println(F("🚿 Bomba LIGADA"));
  digitalWrite(PUMP, HIGH);
  delay(ATTACK_TIME);
  digitalWrite(PUMP, LOW);
  Serial.println(F("💧 Bomba DESLIGADA"));
  // pequeno recuo para se afastar do fogo
  atras(SPEED_TURN); delay(250);
  StopMotores();
}

// ===================== Leitura com média =====================

void readFlames(int &L, int &C, int &R) {
  long sl=0, sc=0, sr=0;
  const int N=6;
  for (int i=0;i<N;i++){
    sl += analogRead(FL_LEFT);
    sc += analogRead(FL_CENTER);
    sr += analogRead(FL_RIGHT);
    delay(2);
  }
  L = sl / N;
  C = sc / N;
  R = sr / N;
}
```