```
// FILE: maze1_0a.nxc - Works!
// DATE: 11/21/22 10:58
// AUTH: P.Oh
// DESC: Domabot maze-solving.  Wall-following with port-side (Port 4) and
//       Obstacle-avoiding with bow-side (Port 1) ultrasonic sensors (US)
//       Essentially combines wfPid1_0a.nxc and oaPid1_0a.nxc algorithms
//       but uses flags for zeroing gains to avoid derivative kick and
//       integral windup during turns.  PID gains for WF and OA are given
//       and were tuned to Paul Oh's Domabot
// VERS: 1.0a: ME 425 release (a copy of wf2Us0_2a4.nxc but with maze1_0a.nxc
//       filename)
// REFS: wfbb0_1a3.nxc, wf2Us0_2a4.nxc

// Global variables ----------------------------------------------------------
  int xWall, xObst;         // wall and obstacle distance [cm]
  int dWall, dObst;         // desired wall and obstacle distances [cm]
  float wKp, wKi, wKd;      // wall PID gains;
  float oKp, oKi, oKd;      // obstacle PID gains;
  float oCorr;              // obstacle related motor power correction [0, 100]
  float wE, wEDot, wEInt;   // wall error and its derivative and integral
  float wEPrev;             // wall error previous value
  float wCorr;              // wall related motor power correction [0, 100]
  float oE, oEDot, oEInt;   // obstacle error and its derivative and integral
  float oEPrev;             // obstacle error previous value
  int speedA, speedC;       // motor speed for A (right) and C (left)
  int speedBase;            // motor base speed
  bool rightTurnBegin;      // Domabot began right (CW) yaw to avoid obstacle
  bool leftTurnBegin;       // Domabot began left (CCW) yaw around corner
  bool straightLineBegin;   // Domabot began wall-following
  bool orangeButtonPushed, rightButtonPushed, leftButtonPushed; // NXT buttons

task main() {

  // Variable initializations ------------------------------------------------
  xWall = xObst = 0;        // initialize wall and obstacle distance to 0
  dWall = 10;               // Desired distance from wall [cm]
  dObst = 30;               // Desired distance to obstacle [cm]
  wKp = 1.5;                // Wall P gain e.g. (PID)=[1.5, 0.005, 30.0] OK
  wKi = 0.005;              // Wall I gain e.g. 0.003
  wKd = 30.0;               // Wall D gain e.g. 8
  oKp = 20.0;               // Obstacle P gain e.g. 40
  oKi = 0.01;               // Obstacle I gain e.g. 0
  oKd = 0.5;                // Obstacle D gain e.g. 20
  wE = wEDot = wEInt = 0.0; // initialize wall error-related values to 0
  oE = oEDot = oEInt = 0.0; // initialize obstacle error-related values to 0
  wEPrev = oEPrev = 0.0;    // initialize previous wall and obstacle errors to 0
  speedBase = 50;           // Domabot base motor speed at 40% i.e. ~mid-point
  rightTurnBegin = leftTurnBegin = straightLineBegin = FALSE; // initially FALSE

  // Algorithm begins --------------------------------------------------------
  TextOut(0, LCD_LINE2, "-> BTN to proceed" );
  SetSensorLowspeed(IN_4); // Wall on Left US (Port 4)
  SetSensorLowspeed(IN_1); // Obstacle in Front US (Port 1)

  do {
    rightButtonPushed = ButtonPressed(BTNRIGHT, FALSE);
    xWall = SensorUS(IN_4); // for wall detection (on left, Port 4)
    xObst = SensorUS(IN_1); // for object detection (front, Port 1)
    TextOut(0, LCD_LINE3, FormatNum("Wall = %3d cm" , xWall));
    TextOut(0, LCD_LINE4, FormatNum("Obst = %3d cm" , xObst));
  } while(!rightButtonPushed);

  ClearLine(LCD_LINE2);
```

```
  TextOut(0, LCD_LINE2, "<- BTN to QUIT" );
  xWall = SensorUS(IN_4); // sense wall distance again

  do { // continue wall following until left button pushed
    leftButtonPushed = ButtonPressed(BTNLEFT, FALSE);
    xObst = SensorUS(IN_1);     // sense obstacle distance
    TextOut(0, LCD_LINE4, FormatNum("Obst = %3d cm" , xObst));

    // (1) Check for obstacle and calculate PID control for sharp CW yaw
    if( xObst <= dObst) {       // Domabot close to obstacle.  Begin right turn
        PlayTone(TONE_A7, 1);
        // (1A) If this is the first time avoiding obstacle, then clear gains
        if(rightTurnBegin == FALSE) {
           // Domabot will now begin avoiding obstacle
           // so avoid derivative kick and integral windup by
           // zeroing the related obstacle errors

           ADD CODE TO ZERO OUT OA PID GAINS  //
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

        }; // end if(rightTurnRight == FALSE)

        // (1B) Calculate PID control effort to avoid obstacle
        oE = xObst - dObst;     // obstacle error = obstacle dist - desired dist
        oEDot = oE - oEPrev;    // obstacle error derivative
        oEInt = oE + oEInt;     // obstacle error integral
        oCorr = oKp * oE + oKi * oEInt + oKd * oEDot;
        if(oCorr > 0 && oCorr > speedBase) oCorr = speedBase;   // saturated
        if(oCorr < 0 && oCorr < -speedBase) oCorr = -speedBase; // saturated
        // (1C) Turning right means CW yaw so motor speed C > motor speed A
        speedC = speedBase - oCorr;
        speedA = oCorr - speedBase;
        OnFwd(OUT_C, speedC);
        OnFwd(OUT_A, speedA);
        // (1D) Domabot has turned (and turning) right i.e. CW yaw

        ADD CODE TO SET FLAGS TRUE, FALSE ETC //
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

    }; // end if(xObst <= dObst)

    // (2) No obstacle, so follow wall.  Calculate PID control correction
    xWall= SensorUS(IN_4);      // left US (Port 4)
    TextOut(0, LCD_LINE3, FormatNum("Wall = %3d cm" , xWall));
    // (2A) Since following wall, reset rightTurnBegin flag

    ADD CODE TO SET FLAGS TRUE, FALSE ETC //
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

    // (2B) Check if too close to wall
    if(xWall < dWall) {
       // (2C) If first time following wall e.g. after avoiding obstacle...
       //      then avoid derivative kick and integral windup by zeroing gains
       if(straightLineBegin == FALSE) {

           ADD CODE TO ZERO OUT WF PID GAINS  //
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

       };
       // (2D) Calculate wall-following PID gains
       wE = xWall - dWall;         // wall error = wall dist - desired wall dist
       wEDot = wE - wEPrev;        // wall error derivative
       wEInt = wE + wEInt;         // wall error integral
```

```
      wCorr = wKp * wE + wKi * wEInt + wKd * wEDot;
      if(wCorr > 0 && wCorr > speedBase) {
          wCorr = speedBase;      // saturated so set correction to speedBase
                                  // So Motor A speed max will be 2*speedBase
      };
      if(wCorr < 0 && wCorr < -speedBase) {
          wCorr = -speedBase;     // saturated so set correction to -speedBase
                                  // So Motor C speed min will be 0
      };
      // (2E) Domabot starting straight line, so change flag

      ADD CODE TO SET FLAGS TRUE, FALSE ETC // <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

    }; // end if(xWall < dWall)

    // (3) Check if far from wall
    if(xWall >= dWall) {
      // (3A) First time Domabot turning left (yawing CCW)
      // e.g. Domabot about to turn CCW around a corner
      // then avoid derivative kick and integral windup for wall and obstacle
      if(leftTurnBegin == FALSE) {

          ADD CODE TO ZERO OUT WF AND OA PID GAINS  // <<<<<<<<<<<<<<<<<<<<<<<<<<<<

      };
      // (3B) Calculate related wall-following PID gains
      wE = xWall - dWall;         // wall error = wall dist - desired wall dist
      wEDot = wE - wEPrev;        // wall error derivative
      wEInt = wE + wEInt;         // wall error integral
      wCorr = wKp * wE + wKi * wEInt + wKd * wEDot;
      if(wCorr > 0 && wCorr > speedBase) {
          wCorr = speedBase;      // saturated so set correction to speedBase
                                  // So Motor A speed max will be 2*speedBase
      };
      if(wCorr < 0 && wCorr < -speedBase) {
          wCorr = -speedBase;     // saturated so set correction to -speedBase
                                  // So Motor C speed min will be 0
      };
      // (3C) Domabot started turning left (CCW yaw) e.g. rounding a corner

      ADD CODE TO SET FLAGS TRUE, FALSE ETC // <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

    }; // end if(xWall > = dWall)

    // (4) Command motors
    speedA = speedBase + wCorr;
    speedC = speedBase - wCorr;
    OnFwd(OUT_C, speedC);
    OnFwd(OUT_A, speedA);
    // (4A) Clear obstacle avoidance gains
    oEDot = oEInt = oEPrev = 0.0;
    // (4B) update wall and obstacle errors for next derivative calculation
    wEPrev = wE;
    oEPrev = oE;
  } while( (!leftButtonPushed)  ); // end do-loop

   // (4) User pushed <-- (Left) Button, so exit gracefully
   Off(OUT_AC);
   PlaySound(SOUND_DOUBLE_BEEP);
   Wait(5000);
   StopAllTasks();

} // end main
```