## Hands-on Lab

## Domabot – Wall Docking with Ultrasonic Sensor

A previous lab introduced the Lego Ultrasonic sensor. Docking is a good application to demonstrate an ultrasonic sensor. In docking, the vehicle approaches a target position e.g. a certain distance from a wall. Here, the ultrasonic sensor reports the distance between the vehicle and wall. The concepts in this lab will contrast open-loop and closed-loop speed control with the Domabot for wall-docking.

**Figure**: Technic Part 32138 Double Pin with Axle (left) attaches ultrasonic sensor to Domabot (right)

### Concept 1 – Open-loop Wall-Docking

**Step 1:** Refer to the figure above and connect the Ultrasonic Sensor to the Domabot. Use Input Port 4 to connect the NXT cables. Click `File – New`. Click `File – Save As` and save in a directory file name "**`us0_1a1.nxc`**".

**Step 2:** Enter the following text

```
// FILE: us0_1a1.nxc - Works!
// DATE: 09/17/22 16:39
// AUTH: P.Oh
// DESC: Domabot proportionally slows down as it gets closer to wall
// REFS: 0.1a: testing ultrasonic sensor (attached to IN_4)
//       0.1a1: Domabot open-loop, moving until desired distance from wall

task main() {

  // variable declarations ------------------------------------------------
  bool orangeButtonPushed, rightButtonPushed, leftButtonPushed; // NXT buttons

  byte distFromWall;                // [0, 255] US sensor value in [cm]
  byte distFromWallDesired;         // [cm] desired distance from wall
  byte distFromWallError;           // [cm] error
  byte threshold;                   // [cm] wiggle room that >= 0

  byte motorSpeed;                  // [0, 100] percent of motor's maximum speed
```

```
  // variable initiations ------------------------------------------------
  motorSpeed = 50; // tune this.  Too fast and Domabot may overshoot
  threshold = 1; // wiggle room i.e. so Domatbot within 19 to 20 cm from wall

  // Beginning program ---------------------------------------------------
  TextOut(0, LCD_LINE2, "-> BTN to proceed" );
  SetSensorLowspeed(IN_4); // US sensor in Port 4
  do {
    rightButtonPushed = ButtonPressed(BTNRIGHT, FALSE);
    distFromWall = SensorUS(IN_4);
    TextOut(0, LCD_LINE3, FormatNum("US= %3d cm" , distFromWall));
  } while(!rightButtonPushed);

  ClearScreen();
  distFromWallDesired = 20; // want Domabot to stop 20 cm from wall
  OnFwd(OUT_AC, motorSpeed); // start moving toward wall

  TextOut(0, LCD_LINE1, "<- BTN to QUIT" );
  do { // continue until left button pushed or within desired distance to wall
    leftButtonPushed = ButtonPressed(BTNLEFT, FALSE);
    distFromWall = SensorUS(IN_4);
    if(distFromWall >= 255) distFromWall = 255; // distance is saturated. Set to max
    distFromWallError = abs(distFromWall - distFromWallDesired); // abs in case too close to wall
    TextOut(0, LCD_LINE3, FormatNum("Dist =%3d" , distFromWall) );
    TextOut(0, LCD_LINE4, FormatNum("Error=%3d" , distFromWallError) );
    TextOut(0, LCD_LINE6, FormatNum("Pwr  =%3d" , motorSpeed) );
    OnFwd(OUT_AC, motorSpeed);
  } while( (!leftButtonPushed) && (distFromWallError >= threshold) )

  // User pushed Left Arrow button to quit or Domabot at desired wall distance
  // So exit gracefully
  Off(OUT_AC);
  PlaySound(SOUND_DOUBLE_BEEP);
  Wait(SEC_10); // long delay so user can read distance and error values in NXT
  StopAllTasks();

} // end main
```

**Figure 1-1**: **`us0_1a1.nxc`** cost for open-loop docking. The Domabot translates at a constant speed until it's at desired distance from the wall

**Step 3:** Click `File – Save All` and then `Compile`.

**Step 4:** Place your Domabot a distance (e.g. 60 cm = 2-feet) from a (solid) wall. If handy, use a measuring tape to mark this initial starting distance. Execute us0_1a1.nxc and observe the Domabot's constant speed. When the Domabot stops, verify the distance of the Domabot's ultrasonic sensor and the wall is within the desired distance. Is there overshoot or undershoot?

**Code Explanation**: The yellow-highlights simply is "Best Practice" where the commented lines to partition code, making it more reader-friendly. First, variables are declared. Second, the variables are initialized with values. Next, the header shows where the code begins operationally. The green-highlight shows the `motorSpeed` value. This is the speed the Domabot travels (constantly) towards the wall. The do-while loop calculates `distFromWallError`, which is the difference between the Domabot's current distance (as measured by the ultrasonic sensor) and the desired distance (i.e. `distFromWallDesired`). One the error is within a threshold value (i.e. 19 to 21 cm), the Domabot stops.

**Exercise 1:** In NxC create programs for the following:

1-1 Change the value of motorSpeed in us0_1a1.nxc. Observe any overshoot (i.e. Domabot is closer to the wall than desired) or undershoot (Domabot is still a bit far from the wall as desired).

**Concept 2 – Closed-loop Docking:** Proportional (closed-loop) control is used. Here, the Domabot's speed slows down (proportionally) as it gets closer to the wall.

**Step 1:** Open a new file and save as "`us0_1d.nxc`". Type the following and save

```
// FILE: us0_1d.nxc - Works!
// DATE: 09/17/22 16:55
// AUTH: P.Oh
// DESC: Domabot proportionally slows down as it gets closer to wall
// REFS: 0.1a: testing ultrasonic sensor (attached to IN_4)
//       0.1a1: Domabot open-loop control.  Translates at constant speed
//       0.1b: Domabot translates towards wall using proportional control
//       0.1b1: test min power for Domabot to move forward.  Ans: 20% on carpet; 10% can still
move on uncarpeted floor
//       0.1c: same as 0.1b but with better Kp gains (observed overshoot)
//       0.1d: cleaned up 0.1c.  Release for ME 425/625

task main() {

  // variable declarations ----------------------------------------------------
  bool orangeButtonPushed, rightButtonPushed, leftButtonPushed; // NXT buttons

  byte distFromWall;                  // [0, 255] US sensor value in [cm]
  byte distFromWallDesired;           // [cm] desired distance from wall
  byte distFromWallError;             // [cm] error
  byte threshold;                     // [cm] wiggle room that >= 0

  byte motorSpeed;                    // [0, 100] percent of motor's maximum speed
  byte motorSpeedMax, motorSpeedMin;  // max and min motor speed as percentage
  float kP;                           // proportional gain
  float floatMotorSpeed;              // float version of motorSpeed

  // variable initiations -----------------------------------------------------
  motorSpeedMax = 70;
  motorSpeedMin = 20;
  kP = 1.5; // Value to tune e.g. 0.5 to 2.0; depends on floor, rug, etc
  threshold = 1; // wiggle room i.e. so Domatbot within 19 to 20 cm from wall

  // Beginning program --------------------------------------------------------
  TextOut(0, LCD_LINE2, "-> BTN to proceed" );
  SetSensorLowspeed(IN_4); // US sensor in Port 4
  do {
    rightButtonPushed = ButtonPressed(BTNRIGHT, FALSE);
    distFromWall = SensorUS(IN_4);
    TextOut(0, LCD_LINE3, FormatNum("US= %3d cm" , distFromWall));
  } while(!rightButtonPushed);

  ClearScreen();
  distFromWallDesired = 20; // want Domabot to stop 20 cm from wall
```

```
  TextOut(0, LCD_LINE1, "<- BTN to QUIT" );
  do { // until left button pushed or within desired wall distance
    leftButtonPushed = ButtonPressed(BTNLEFT, FALSE);
    distFromWall = SensorUS(IN_4);
    if(distFromWall >= 255) distFromWall = 255; // distance is saturated. Set to max
    distFromWallError = abs(distFromWall - distFromWallDesired); // abs in case too close to wall
    TextOut(0, LCD_LINE3, FormatNum("Dist =%3d" , distFromWall) );
    TextOut(0, LCD_LINE4, FormatNum("Error=%3d" , distFromWallError) );
    floatMotorSpeed = kP * distFromWallError; // proportional speed control
    motorSpeed = floatMotorSpeed; // forces floatMotorSpeed into a byte
    if(motorSpeed >= motorSpeedMax)
       motorSpeed = motorSpeedMax; // Speed saturated.  Set to Max
    if(motorSpeed <= motorSpeedMin)
       motorSpeed = motorSpeedMin; // Speed saturated.  Set t Min
    TextOut(0, LCD_LINE6, FormatNum("Pwr  =%3d" , motorSpeed) );
    OnFwd(OUT_AC, motorSpeed);
  } while( (!leftButtonPushed) && (distFromWallError >= threshold) )

  // User pushed Left Arrow button to quit or Domabot at desired wall distance
  // So exit gracefully
  Off(OUT_AC);
  PlaySound(SOUND_DOUBLE_BEEP);
  Wait(SEC_10); // long delay so user can read distance and error values in NXT
  StopAllTasks();

} // end main
```

**Figure 2-1**: **us0_1d.nxc** code for closed-loop proportional speed wall-docking

**Step 2**: Once successful compiled, place your Domabot some distance (e.g. 60 cm = 2-feet) from a (solid) wall. Execute **us0_1d.nxc**. Observe the Domabot's speed as it approaches the wall. The speed should continually slow down as it gets closer to the wall. Once the Domabot reaches the desired distance from the wall, it should stop.

**Code Explanation**: The code for us0_1d.nxc is very similar to us0_1a1.nxc. The yellow-highlight shows that the Domabot's speed (floatMotorSpeed) is calculated. The speed uses a constant gain kP to multiply the error distFromWallError. Since kP is a float, we save the product into a real variable floatMotorSpeed. Unlike ANSI C which has typecasting, we equate motorSpeed with floatMotorSpeed to force motorSpeed to be byte. One should force such typecasting because of OnFwd. The NXC manual states that OnFwd takes a byte variable (and not a float variable).

**Exercise 2:** In NxC create programs for the following:

2-1 Use different values of kP in us0_1d.nxc. For example try 0.5 and 2.0. What do you observe in the time it takes Domabot to reach the desired wall distance? What do you observe in overshoot or undershoot?

2-2 Use different values of motorSpeed in us0_1d.nxc. For example try 30 and 90. What do you observe in the time it takes Domabot to reach the desired wall distance? What do you observe in overshoot or undershoot?