```
// FILE: xl320-ik-1_0.nxc - Works!
// DATE: 01/16/20 09:12
// AUTH: P.Oh
// DESC: Inverse kinematics for 2-DOF planar manipulator using Dynamixel XL-320
// VERS: 0.1a: use H-files
// REFS: twoLinkDynamixelXl320-2.0a
// NOTE: If factory default XL-320 used, then ID is 0x01
//       ID of 0xFE commands any and all XL-320 motors
//       This example uses an XL-320 configured with ID# 3

// #include "DynamixelXL320.h"

#include "xl320-defines1_0a.h"   // XL-320 defines from Control Table
#include "xl320-functions1_0d.h" // P.Oh functions written for XL-320

#define ID_ALL_MOTORS 0XFE // 0XFE commands all XL-320 motors
#define ID_MOTOR01    0X03 // Assumes Motor 1 configured with ID = 3
#define ID_MOTOR02    0X07 // Assumes Motor 2 configured with ID = 7
#define mmPerStud     8    // 8 millimeters per LEGO stud

// Global variables
  bool  orangeButtonPushed;            // Detect Brick Center button state
  bool  rightArrowButtonPushed;        // Detect Brick right arrow button state

void rotateMotorAbsolutely(float angle01, float angle02) { //------------------
 // Rotates desired the two Dynamixel XL-320 motors to their desired angles
 // Assumes motor count of 512 denotes 0 degrees.  Uses right-hand rule for
 // rotational direction

  float desiredAngle01InDegrees;       // Angle Motor 1 to move to [deg]
  float desiredAngle02InDegrees;       // Angle Motor 2 to move to [deg]
  float degreesPerCount;               // Conversion 0.29 [degrees/count]
  float calculatedCount;               // Count equivalent of desired angle [count]
  int   motor01Offset;                 // Motor 1's offset [count]
  float theta01InDegrees;              // Motor 1 angle [counts]
  int   theta01InCounts;               // Motor 1 angle [deg]
  int   motor02Offset;                 // Motor 2's offset [count]
  float theta02InDegrees;              // Motor 2 angle [counts]
  int   theta02InCounts;               // Motor 2 angle [deg]
  string msg01, msg02;                 // dummy strings to print values to screen

  motor01Offset = 512; // Set Link 1 at 0 deg (i.e. 512 counts)
  motor02Offset = 512; // Set Link 2 at 0 deg (i.e. 512 counts)

  // Note 1: Looking into horn from Top, count > 512 is CCW (i.e. +Z axis)
  // and count < 512 is CW (i.e. -Z axis)
  degreesPerCount = 0.29; // [deg/count] found from XL-320 data sheet

  ClearScreen();
  desiredAngle01InDegrees = angle01;
  theta01InCounts = motor01Offset + desiredAngle01InDegrees/degreesPerCount;
  desiredAngle02InDegrees = angle02;
  theta02InCounts = motor02Offset + desiredAngle02InDegrees/degreesPerCount;

  // Format string so displays nicely on Brick screen
  sprintf(msg01, "Go [%3.1f, " ,desiredAngle01InDegrees);
  sprintf(msg02, "%3.1f]" , desiredAngle02InDegrees);
  TextOut(0, LCD_LINE2, strcat(msg01, msg02));

  XL320_servo(ID_MOTOR01, theta01InCounts, 200); // motor position at speed 200
  Wait(2000); // wait about 2 seconds before issuing another command
  XL320_servo(ID_MOTOR02, theta02InCounts, 200); // motor position at speed 200
  Wait(2000); // wait about 2 seconds before issuing another command
```

```
  PlayTone(TONE_B3,50);

}; // end rotateMotorAbsolutely function --------------------------------

task main() {

  // planar manipulator variables
  float l1, l2; // length of link 1 and link 2 [mm]
  float theta1, theta2; // angle of joint 1 and joint 2 [rad]
  float theta1InDegrees, theta2InDegrees; // angle of joint 1 and 2 [deg]
  float xCalibrate[5], yCalibrate[5]; // four (x,y) calibration points wrt x0y0
frame [mm]
  ArrayInit(xCalibrate, 0, 5); // initialize the (4x1) x vector with zeros
  ArrayInit(yCalibrate, 0, 5); // initialize the (4x1) y vector with zeros
  float xP, yP; // end-effector absolute position i.e. wrt x0y0 frame [mm]

  // calculation and dummy variables
  float C, k1, k2, num, den;
  int i;

  // initializations
  l1 = 7 * mmPerStud; // [mm] link 1 is 7 studs long
  l2 = 5 * mmPerStud; // [mm] link 2 is 5 studs long
  // xCalibration[i] and yCalibrate[i] in [mm]
  // 90-degree calibration points easiest to envision end-effector location
  xCalibrate[0] = l1; yCalibrate[0] = l2;  // +'ve root: (theta1, theta2)=(0,90)
  xCalibrate[1] = l1; yCalibrate[1] = -l2; // -'ve root: (theta1, theta2)=(0,-90)
  xCalibrate[2] = l2; yCalibrate[2] = l1;  // -'ve root: (theta1, theta2)=(90,-90)
  xCalibrate[3] = l2; yCalibrate[3] = -l1; // +'ve root: (theta1, theta2)=(-90,90)
  // slightly harder to envision example
  // [mm] (theta1, theta2) will = +'ve root (8.7, 43.2) or -'ve (44.4, -43.2)
  xCalibrate[4] = 10 * mmPerStud; yCalibrate[4] = 5 * mmPerStud;

  UseRS485();
  RS485Enable();
  RS485Uart(HS_BAUD_57600, HS_MODE_8N1); //57600 baud, 8bit, 1stop, no parity

  ClearScreen();
  // Prompt user to begin
  TextOut(0, LCD_LINE1, "Start: hit ->");
  do {
     rightArrowButtonPushed = ButtonPressed(BTNRIGHT, FALSE);
  } while(!rightArrowButtonPushed);
  ClearScreen();

  // First go to home position
  ClearScreen();
  TextOut(0, LCD_LINE2, "Homing..." );
  Wait(2000);
  theta1InDegrees = theta2InDegrees = 0.0;
  rotateMotorAbsolutely(theta1InDegrees, theta2InDegrees);
  Wait(2000);
  PlayTone(TONE_E4, 500);

  // Next, go to desired points
  for(i=0; i<=4; i++) { // cycle thru the 4 calibration points and 5th point
      xP = xCalibrate[i];
      yP = yCalibrate[i];
      // pow function for power.  Using ^ is incorrect
      C = ( pow(xP,2)+pow(yP,2) - pow(l1,2)-pow(l2,2) ) / (2*l1*l2);
      if(i==0 || i==3) { // choose +'ve root
         num = sqrt(1-pow(C,2));
      } else { // use -'ve root for xPCalibrate[1], yPCalibrate[1] theta2 should be
```

```
-90
        num = sqrt(1-pow(C,2));
      };
      theta2 = atan2(num, C); // [rad]
      theta2InDegrees = theta2 * 180/PI; // [deg]
      k1 = l1 + l2*cos(theta2);
      k2 = l2*sin(theta2);
      theta1 = atan2(yP, xP) - atan2(k2, k1); // [rad]
      theta1InDegrees = (theta1 * 180/PI); // [deg]

      // Actuate the XL-320 motors
      rotateMotorAbsolutely(theta1InDegrees, theta2InDegrees);
  }; // end for-loop

  // Go back to home position
  ClearScreen();
  TextOut(0, LCD_LINE2, "Back to Home" );
  Wait(2000);
  theta1InDegrees = theta2InDegrees = 0.0;
  rotateMotorAbsolutely(theta1InDegrees, theta2InDegrees);
  Wait(2000);
  PlaySound(SOUND_DOUBLE_BEEP);
} // end main
```