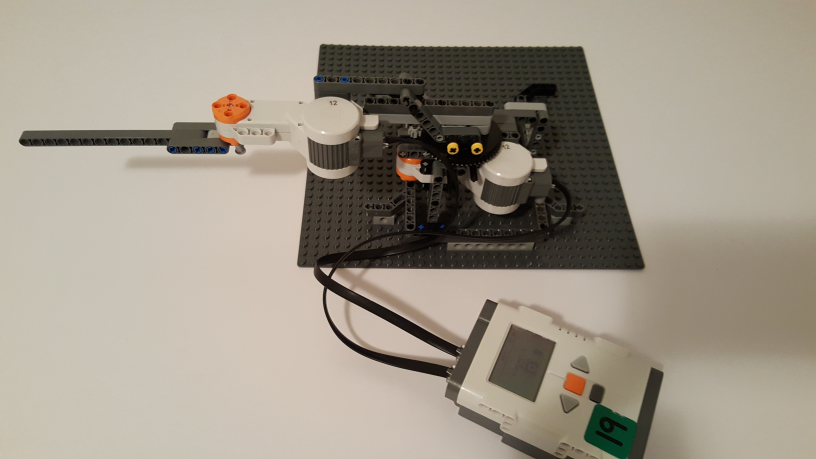
**Hands-on Lab**

**Lego Control – Forward Kinematics**

Robot manipulator joints are typically revolute or prismatic. Denavit-Hartenberg (DH) prescribes an analytical method to naming and locating reference frames with an -link manipulator. This lab will construct a 2-link (revolute) planar manipulator. This manipulator will be used to better understand forward kinematics – calculate the end-effector position for given joint variables.

**Concept 1 – Construction**



**Fig. 1A:** A planar 2-link (revolute joints) manipulator

Refer to the NXT build plan (lego2dPlanarArmBuildPlan-103016.pdf) to construct Fig. 1A

Step 2: Button Example – the NXT only has 3 controllable buttons (the 4th is the grey Exit button, which aborts execution). Write the following program **button1\_0.nxc** to understand the role of ButtonPressed and ButtonCount

// FILE: button1\_0.nxc - Works!

// DATE: 10/30/16 12:57

// AUTH: P.Oh

// VERS: 1.0: Detect and display number of times a button is pushed

task main() {

// button variables

bool orangeButtonPushed, rightArrowButtonPushed, leftArrowButtonPushed;

bool greyButtonPushed;

int countOrangeButton, countRightArrowButton, countLeftArrowButton;

// initializations

countOrangeButton = countRightArrowButton = countLeftArrowButton = 0;

// Prompt user to begin

PlayTone(TONE\_B3,50);

TextOut(0, LCD\_LINE1, "Grey BTN Quits");

do {

greyButtonPushed = ButtonPressed(BTNEXIT, FALSE);

countOrangeButton = ButtonCount(BTNCENTER, FALSE);

countRightArrowButton = ButtonCount(BTNRIGHT, FALSE);

countLeftArrowButton = ButtonCount(BTNLEFT, FALSE);

TextOut(0, LCD\_LINE3, FormatNum("Orange: %d", countOrangeButton));

TextOut(0, LCD\_LINE4, FormatNum("Left: %d", countLeftArrowButton));

TextOut(0, LCD\_LINE5, FormatNum("Right: %d", countRightArrowButton));

} while(!greyButtonPushed);

} // end main

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

* 1. Write button1\_1.nxc that counts and displays the number of times the Orange button is pushed. If the count is even, then display that the count number is even and likewise if the count is odd.

**Concept 2 – Open-loop control (command joint variables)**

NXC has various functions to command the NXT motors. These are explored to contrast them.

Step 1: Write the program planar1\_0.nxc to test your 2-link planar manipulator.

// FILE: planar1\_0.nxc - works!

// DATE: 10/29/16 17:39; 10/30/16 12:36

// AUTH: P.Oh

// VERS: 1.0: 2-DOF planar robot test

// NOTE: Port A (DOF 1), Port B (DOF 2)

// BLAH: Next time: deal with relative vs. absolute angles...

task main() {

// button variables

bool orangeButtonPushed, rightArrowButtonPushed, leftArrowButtonPushed;

bool greyButtonPushed;

// gear variables

int turnTableTeeth = 56; // For DOF 1, which has the gear ratio

int spurTeeth = 8; // NB: +Angle yields CCW rotatio (with eye on top of turntable)

float gearRatio = 7;

float desiredTurnTableAngleInDegrees; // [deg]

float spurAngleInDegrees; // [deg]

// Prompt user to begin

PlayTone(TONE\_B3,50);

TextOut(0, LCD\_LINE1, "Rotate motors");

TextOut(0, LCD\_LINE2, "to set HOME");

TextOut(0, LCD\_LINE3, "Then hit ->");

do {

rightArrowButtonPushed = ButtonPressed(BTNRIGHT, FALSE);

} while(!rightArrowButtonPushed);

// Motors are in HOME position. Set their current angles to 0

PosRegEnable(OUT\_A); // Set Port A (DOF 1) current angle as zero [deg]

PosRegSetMax(OUT\_A, 40, 0); // Set Port A speed limit (40) and default acceleration (0)

PosRegSetAngle(OUT\_A, 0);

PosRegEnable(OUT\_B); // Set Port B (DOF 2) current angle as zero [deg]

PosRegSetAngle(OUT\_B, 0);

PosRegSetMax(OUT\_B, 40, 0); // Set Port B speed limit (40) and default acceleration (0)

ClearScreen();

PlayTone(TONE\_B3,50);

TextOut(0, LCD\_LINE1, "DOF1 +90 = CCW");

TextOut(0, LCD\_LINE2, "Hit ORG to start");

do {

rightArrowButtonPushed = ButtonPressed(BTNCENTER, FALSE);

} while(!rightArrowButtonPushed);

desiredTurnTableAngleInDegrees = 90;

spurAngleInDegrees = desiredTurnTableAngleInDegrees \* gearRatio;

// RotateMotor(OUT\_A, 50, spurAngleInDegrees);

// Could replace RotateMotor with PosRegAddAngle(OUT\_A, spurAngleInDegrees);

// But PosRegAddAngle has some overshoot due to default gains on motor

// PosRegAddAngle(OUT\_A, spurAngleInDegrees);

PosRegSetAngle(OUT\_A, spurAngleInDegrees);

Wait(5000);

PlayTone(TONE\_B3,500);

ClearScreen();

PlayTone(TONE\_B3,50);

TextOut(0, LCD\_LINE1, "DOF1 -90 = CW");

TextOut(0, LCD\_LINE2, "Hit ORG to start");

do {

rightArrowButtonPushed = ButtonPressed(BTNCENTER, FALSE);

} while(!rightArrowButtonPushed);

desiredTurnTableAngleInDegrees = -90;

spurAngleInDegrees = desiredTurnTableAngleInDegrees \* gearRatio;

// RotateMotor(OUT\_A, 50, spurAngleInDegrees);

// PosRegAddAngle(OUT\_A, spurAngleInDegrees);

PosRegSetAngle(OUT\_A, spurAngleInDegrees);

Wait(5000);

PlayTone(TONE\_B3,500);

} // end main

Code Explanation: Degree-of-freedom 1 (DOF 1) is Motor 1 which is connected to Port A. This has an 8 tooth spur gear that meshes with a 56 tooth turntable gear. PosRegEnable is used to establish the desired motor position as 0 degrees.

The program planar1\_0.nxc uses button presses to determine when the user has defined the manipulator’s home position (both angles are then defined as zero). The gear ratio (56:8) is used to determine the number of times the spur must rotate.

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

* 1. Uncomment lines in planar1\_0.nxc to contrast PosRegEnable(OUT\_A) and PosRegEnable(OUT\_A, 40, 0). Try different values to contrast response rate and overshoot

2-1 Uncomment lines in planar1\_0.nxc to contrast RotateMotor, PosRegAddAngle,

PosRegSetAngle. What are the differences?

* 1. Write an NXC program planar1\_1.nxc that does the following:
* If Orange button count is even, then switch degrees-of-freedom
* Left button moves DOF 45 deg CCW. Right button moves DOF 45 deg CW