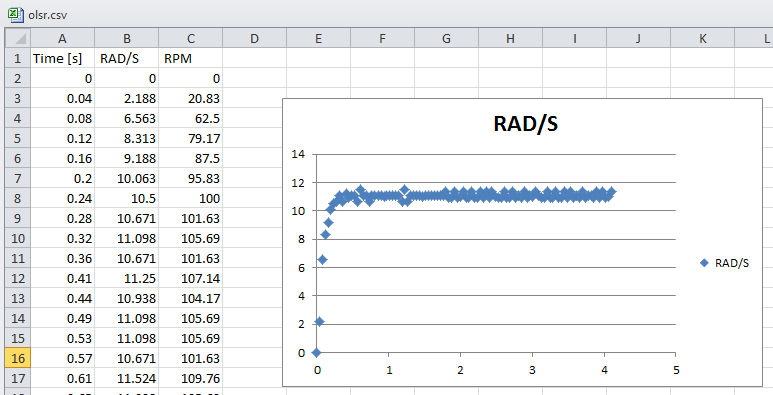
**Hands-on Lab**

**Lego NXT – NXT Motor Open-Loop Step Response (OLSR)**

In class, the theory underlying a DC motor was presented. Briefly, a DC motor can be modeled as a first order system. Here, a step input (i.e. a constant voltage) is applied to the motor. The motor’s rotational speed gradually reaches a steady-state value as depicted in **Figure 1** below. This lab demonstrates this phenomenon. CountTick is used to set the sampling time, and to calculate the time between changes in the motor’s angle. The resulting speed (rad/s and RPM) are computed and saved as a file. XLS scatter plot (Figure 1 graph) shows the motor’s first-order response.



**Figure 1**: Open-Loop Step Response of a LEGO NXT motor

**Step 1**: Write save, and compile **motorOlsr1\_0a.nxc**

Note: The program is long only because it is filled with comments. Moreover, the code leverages the concepts learned in file handling (e.g. x^2File1.0.nxc) and timers (e.g. stopWatch1\_0a.nxc).

// File: displaySquareAndSquareRoot3\_0.nxc

// FILE: mtrOlsr1\_0a.nxc - Works!

// AUTH: P.Oh

// DATE: 03/13/23 15:58

// DESC: OLSR of NXT motor (Port A) with data written to file olsr.csv

// with 40 msec sampling time

// VERS: 1\_0a: ME 425/625 Release version

// REFS: nxtMotorOlsr1\_0.nxc; x^2File1.0.nxc, rotate0\_1a.nxc, mtrSpeed0\_2a5.nxc

// mtrOlsr0\_1a.nxc

// NOTE: Uses MotorRotationCount which reports encoder count in degrees

// and program calculates difference over delta tic counts

#define MOTOR OUT\_A // set constant MOTOR for Port A

#define FULL\_SPEED 75 // 75 percent of possible motor speed

#define DEG2RPM 166.667 // deg/msec to RPM

#define RPM2RADPERSEC 0.105; // RPM to rad/sec

task main() {

// Declare variable -----------------------------------------------------

// Motor related variables

long degPrev, degCurr, degDelta; // motor's degrees previous,current, delta

float motorRpm; // motor speed [RPM]

string strMotorRpm; // string form of motorRpm

float motorRadPerSec; // motor speed [rad/s]

string strMotorRadPerSec; // string form of motorRadPerSec

// File related variables

unsigned int result; // flag returned when handling files

byte fileHandle; // handle to the data file

short bytesWritten; // number of bytes written to the file

string fileName, fileHeader, text; // name, header and text to write to file

// Timing related variables

long ticPrev, ticCurr, ticDelta; // previous, current and delta ticks

long ticWait, ticEnd; // ticks to wait and to end

long msSamplingTime; // sampling time in [msec]

float secElapsed; // seconds elapsed in [sec]

string strSecElapsed; // string form of secElapsed

// Button related variables

bool orangeButtonPushed, rightArrowButtonPushed;

// Initialize variables --------------------------------------------------

secElapsed = 0.0; // set elapsed time to zero

degPrev = 0; // motor initially motionless so set angle to zero

msSamplingTime = 40; // [msec] sampling time

// Algorithm starts here ---------------------------------------------------

// (1) Set up the file

fileName = "olsr.csv" ; // <---- file name you want data saved to

result=CreateFile(fileName, 2048, fileHandle);

// (1A) Check if filename already exists, and overwrite it

while (result==LDR\_FILEEXISTS) {

CloseFile(fileHandle);

DeleteFile(fileName);

// result=CreateFile(fileName, 1024, fileHandle);

result=CreateFile(fileName, 2048, fileHandle);

} // end while

// (1B) write column header to file

fileHeader = "Time [s], RAD/S, RPM" ; // <---- column header in your CSV file

WriteLnString(fileHandle, fileHeader, bytesWritten);

// Begin step response

TextOut (0, LCD\_LINE1, "Orange Btn quits" );

strSecElapsed = FormatNum("%5.3f" , secElapsed);

// (2A) read current motor angle [deg] and poll timer before applying power

degPrev = MotorRotationCount(MOTOR);

// Command motor to move i.e. step input

ticPrev = CurrentTick(); // poll timer 1st time <<<<<<<<<<<<<<<<<<<<<<<<<<<

OnFwd(MOTOR, FULL\_SPEED); // turn on motor at FULL\_SPEED value

do {

// (2C) poll timer a 2nd time, measure elapsed time in [msec], read

// motor's rotational angle [deg] and compute rotational speed [RPM]

ticCurr = CurrentTick(); // poll timer a 2nd time <<<<<<<<<<<<<<<<<<<<<<<<<

// Read change in motor angle

degCurr = MotorRotationCount(MOTOR); // get relative position [deg]

degDelta = degCurr - degPrev;

// Measure elapsed time and hence motor RPM

ticDelta = ticCurr - ticPrev; // time [msec] elapsed between angle reads

motorRpm = (degDelta \* DEG2RPM) / ticDelta; // deg/ms \* DEG2RPM yields RPM

strMotorRpm = FormatNum("%5.2f" , motorRpm);

// Display motor actual speed and elapsed time

TextOut(0, LCD\_LINE4, FormatNum("RPM = %5.2f" , motorRpm));

TextOut(0, LCD\_LINE6, FormatNum("Time = %5.3f s" , secElapsed));

// (2D) compute rotational speed [rad/s]. Form string versions of the data

// and write to file

motorRadPerSec = motorRpm \* RPM2RADPERSEC; // RPM \* RPM2RADPERSEC gives rad/s

strMotorRadPerSec = FormatNum("%5.3f" , motorRadPerSec);

secElapsed = secElapsed + (ticDelta/1000.0); // [sec]

strSecElapsed = FormatNum("%1.2f" , secElapsed);

text=StrCat(strSecElapsed, "," , strMotorRadPerSec, "," , strMotorRpm);

WriteLnString(fileHandle, text, bytesWritten);

// (2E) call the current values the previous ones before looping back

ticPrev = ticCurr; // assign 2nd polled tick value to 1st polled tick value

degPrev = degCurr; // assign current rot'n count as previous rot'n count

// (2F) compute how much time to do nothing until sampling cycle is completed

// EG: suppose the current loop iteration took 35 msec. Suppose the defined

// sampling time is 40 msec. Thus, we should wait for 5 msec before looping

// back.

ticEnd = CurrentTick(); // <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

ticWait = msSamplingTime - (ticEnd - ticCurr);

Wait(ticWait);

// Check if user wants to quit

orangeButtonPushed = ButtonPressed(BTNCENTER, FALSE);

} while( !orangeButtonPushed );

// Orange button pressed, so command 0 speed to motor and quit

ClearScreen();

TextOut(0, LCD\_LINE2, "Quitting", false);

// Stop motor

OnFwd(MOTOR, 0);

CloseFile(fileHandle);

PlaySound(SOUND\_LOW\_BEEP); // Beep to signal quitting

Wait(SEC\_2);

StopAllTasks();

} // end of main

**Program:** **motorOlsr1\_0a.nxc**

**Step 2**: Execute motorOlsr1**\_0a.nxc**. This will immediately have the NXT motor (that’s connected to Port A) move at 75% power. Count 3 to 5 seconds, and press the Orange button. This will stop the motor and quit the program.

**Step 3:** Refer to Concept 1 in the Lab entitled **labBricxxFileHandling-102822a.pdf**. Recall, one can use BrixCC’s Tools- NXT Explorer to see the files in one’s Brick. Click and drag the file olsr.csv to your laptop. Use Excel to open the file and create a scatter plot (like the one shown above in Figure 1).

**Code Explanation**:

Blah: add content to describe the code.

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

* 1. Figure 1A is an example of an open-loop step (velocity) response to a 75% motor command. From the plot, calculate the rise time Recall that rise-time (also called time constant) is defined as 63.3% of steady-state.
  2. Theory states for a first-order system, that at 3 time constants, the response will be within 1% of steady-state. Calculate 3 x rise time and find the velocity at the time. Verify that this velocity is within 1% of the steady-state velocity.
  3. Write a new program that performs an open-loop step response but acquires the NXT motor’s position from 0 to 5 seconds (use a 100 millisecond sample time). Plot the curve. Why does this curve loop like a ramp?