Hands-on Lab

Lego Programming – NxC Motor Velocity and Data Acquisition

This lab continues to add new statements in NxC by focusing on data acquisition. Specific concepts include: monitoring motor velocity and file handling.

Preamble Note: You must use BricxCC version `3.3.8.9` (latest). You also need to install the latest driver for your brick (1.031). See course web site for links for these pieces of software.

Concept 1 – Motor Velocity:

Program the NxC to read the NXT motor's internal encoder to display motor RPM on the NXT Brick.

Step 1: Click `File – New`. Click `File – Save As` and save in a directory e.g. "`myPrograms`" with the name "`helloMotorSpeed1_0`".

Step 2: Enter the following text

```
// FILE: helloMotorSpeed_0.nxc
// AUTH: P.Oh
// DATE: 04/19/11 17:30
// DESC: Read and display Motor speed
// NOTE: NXT motor plugged into Brick Port A
// REFS: See pg. 1996 of NXT Guide Section 9.908
// VERS: Compiled with BricxCC 3.3.8.9 and firmware 01.031
// STAT: Works!

#define MOTOR         OUT_A     // set constant MOTOR for Port A
#define FULL_SPEED    100       // 100 percent of possible motor speed
#define DEG2RPM       166.667   // deg/msec to RPM

task main() {

  long prevAngleInDegrees; // placeholder for degree read by motor encoder
  long curAngleInDegrees; // current motor angle [DEG]
  long deltaAngleInDegrees; // change in motor angle [DEG]

  long prevTick;
  long curTick; // current timer value
  long deltaT; // For calculating time between ticks

  float motorRpm; // motor speed [RPM]
  string strMotorRpm; // store integer value of motorRpm as string
  string strDeltaT;
  string strDeltaAngleInDegrees;

  prevAngleInDegrees = 0; // motor initially motionless so set angle to zero
  prevTick = CurrentTick();

  OnFwd(MOTOR, FULL_SPEED); // turn on motor at 100% full speed
```

```
while(true) {

    curAngleInDegrees = MotorRotationCount(MOTOR); // get relative position
    deltaAngleInDegrees = curAngleInDegrees - prevAngleInDegrees;
    strDeltaAngleInDegrees = FormatNum("deltaAngle = %ld", deltaAngleInDegrees);

    curTick = CurrentTick(); // read timer value
    deltaT = curTick - prevTick; // measure time elapsed between angle reads
    strDeltaT = FormatNum("deltaT = %ld", deltaT);

    // NB: Found bug: need to multiply by a float to force motorRpm to be float
    // If use motorRpm = deltaAngleInDegrees / deltaT * DEG2RPM;
    // then motorRpm will return zero

    motorRpm = deltaAngleInDegrees * DEG2RPM / deltaT;
    strMotorRpm = FormatNum("RPM = %5.1f", motorRpm);

    TextOut(0, LCD_LINE2, strDeltaT);
    TextOut(0, LCD_LINE4, strDeltaAngleInDegrees);
    TextOut(0, LCD_LINE6, strMotorRpm);

    prevTick = curTick;
    prevAngleInDegrees = curAngleInDegrees;

    Wait(MS_500);  // update display every 0.5 seconds

  } // end while
} // end main
```

Program: `helloMotorSpeed1_0`

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

Step 4: Connect an NXT motor to Port A. Execute the program by launching `Compile – Download and Run`. Your NXT brick should display the time between ticks, and motor RPM

Exercise 1: In NxC create programs for the following:

1-1 Execute helloMotorSpeed1_0 and write down the RPM. This corresponds to the speed at 100% power. Change the program to command the motor at 50%. Write down the corresponding RPM. Repeat this to complete the following table:

Power level	RPM
100%	Approximately 140 RPM
75%	
50%	
25%	

Concept 2 – Data Acquisition (file handling):

You have experience reading sensors attached to the Lego NXT brick. You have also exercised the ability to display sensor values on the brick's LCD. NxC also has provides the ability to create data files which are saved in the brick's on-board flash memory. One could then capture and write sensor data to a file. These files can then be downloaded to a PC, viewed and even imported to generate graphs.

This concept will sample the motor's velocity and save to a data file. You will then upload the data file to your PC and plot the data in Excel.

Step 1: Open a new file and save as "`helloWriteMotorSpeedToFile1_0`". Type the following and save

```
/ FILE: helloWriteMotorSpeedToFile1_0.nxc
// AUTH: P.Oh
// DATE: 04/20/11 11:48
// DESC: Command motor to max speed, write encoder values to CSV file
// VERS: Compiles under Bricxcc 3.3. Build 3.3.8.9 and firmware 01.031
// STATS: Works!

  #define MOTOR         OUT_A     // set constant MOTOR for Port A
  #define FULL_SPEED    100       // 100 percent of possible motor speed
  #define DEG2RPM       166.667   // deg/msec to RPM

  // Global variables
  unsigned int result; // flag returned when handling files
  byte fileHandle; // handle to the data file
  short fileSize; // size of the file
  short bytesWritten; // number of bytes written to the file
  string fileHeader; // column header for data in the file
  int fileNumber, filePart; // integers to split up data file names
  string fileName; // name of the file
  string strFileNumber; // file number e.g myDataFile 1, 2, 3
  string strFilePart; // file part e.g. myDataFile1-1, 1-2, 1-3
  string text; // string to be written to file i.e. data values
```

Code Example: Part 1 – setting up global variables for `helloWriteMotorSpeedToFile1_0`

Step 2: We now need to tell NxC that we want to define a data file to save your acquired data. We call this file "`myMotorSpeed.csv`", which is a comma-separate data file (so packages like Excel can easily import). NxC (like in the C language) must be told how big the data file will. In our case, we will limit data files to 1024 bytes. We will also check if there are pre-existing data files; we don't want to overwrite existing ones. Every time a new file is created, the brick will beep. Continue typing in the following code in your NxC program.

```
// Create and initialize a file
void InitWriteToFile() {
  fileNumber = 0;  // set first data file to be zero
  filePart = 0; // set first part of first data file to zero
  fileName = "myMotorSpeed.csv"; // name of data file
  result=CreateFile(fileName, 1024, fileHandle);
  // NXT Guide Section 9.100 pg. 1812 and Section 6.59.2.2 pg. 535
  // returns file handle (unsigned int)

  // check if the file already exists
  while (result==LDR_FILEEXISTS) // LDR_FILEEXISTS is returned if file pre-exists
     {
      CloseFile(fileHandle);
      fileNumber = fileNumber + 1; // if data file already exists, create new one
      fileName=NumToStr(fileNumber);
      fileName=StrCat("myMotorSpeed", fileName, ".csv");
      result=CreateFile(fileName, 1024, fileHandle);
      } // end while

  // play a tone every time a file is created
  PlayTone(TONE_B7, 5);
  fileHeader = "Tick, Motor Speed"; // header for myData file
  WriteLnString(fileHandle, fileHeader, bytesWritten);
  // NXT Guide Section 6.59.2.43 pg. 554
  // Write string and new line to a file
  // bytesWritten is an unsigned int.  Its value is # of bytes written

} // end InitWriteToFile
```

Code Example: Part 2 – a function to initialize a file for `helloWriteMotorSpeedToFile1_0`

Step 3: As one acquires data, 1024 bytes might not be enough. Thus, one should create new data files as needed. This function will use the same file name (`myMotorSpeed`) as a prefix and then append -1, -2,… -n as one acquires data (or the brick's flash memory gets maxed). The next function also tells NxC how to close a file. Continue typing the code below into your NxC program.

```
void WriteToFile(string strTempText) {
  // strTempText stores the text (i.e. ticks and motorRpm to be written to file

  // write string to file
  result=WriteLnString(fileHandle, strTempText, bytesWritten);
  // if the end of file is reached, close the file and create a new part
  if (result==LDR_EOFEXPECTED) // LDR_EOFEXPECTED is flagged when end-of-file
    {
    // close the current file
    CloseFile(fileHandle);
    // NXT Guide Section 6.59.2.1 pg. 535
    // Closes file associated with file handle

    // create the next file name
    filePart = filePart + 1;
    strFileNumber = NumToStr(fileNumber);
    strFilePart = NumToStr(filePart);
    fileName = StrCat("myMotorSpeed", strFileNumber,"-", strFilePart ,".csv");

    // delete the file if it exists
    DeleteFile(fileName);
    // NXT Guide Section 6.59.2.5 pg. 537
    // Delete the file specified by the string input

    // create a new file
    CreateFile(fileName, 1024, fileHandle);
    // play a tone every time a file is created
    PlayTone(TONE_B7, 5);
    WriteLnString(fileHandle, strTempText, bytesWritten);
    } // end if
} // end WriteToFile

// Close the file
void StopWriteToFile() {
  // close the file
  CloseFile(fileHandle);
} // end StopWriteToFile
```

Code Example: Part 3 – a function to write to a file (creating new ones as needed) and for closing files for `helloWriteMotorSpeedToFile1_0`

Step 4: `main` is where all NxC (as in the C language) where programs always start. Variables are first declared. The call `to InitWriteToFile` tells NxC to create and initialize the data file. The motor (connected to Port A) is commanded to move a full speed (using `OnFwd`) and then an endless `while` loop is entered. The `while` loop exits if either the brick's orange button is pressed or the brick runs out of memory. Continue typing the following in your NxC program

```
task main() {

  long prevAngleInDegrees; // placeholder for degree read by motor encoder
  long curAngleInDegrees; // current motor angle [DEG]
  long deltaAngleInDegrees; // change in motor angle [DEG]

  long prevTick;
  long curTick; // current timer value
  long deltaT; // For calculating time between ticks
  float elapsedTimeInSeconds; // time in seconds
  string strElapsedTimeInSeconds; // string representation of elapsed time

  float motorRpm; // motor speed [RPM]
  string strMotorRpm; // store integer value of motorRpm as string
  string strDeltaT; // store value of deltaT as string
  string strDeltaAngleInDegrees; // store value of deltaAngleInDegrees as string

  //Create a new file that captures time and motor speed
  InitWriteToFile();

  prevAngleInDegrees = 0; // motor initially motionless so set angle to zero
  elapsedTimeInSeconds = 0.0; // set elapsed time to zero

  prevTick = CurrentTick();

  OnFwd(MOTOR, FULL_SPEED); // turn on motor at 100% full speed

  // Never ending loop until the orange button is pressed
  // or the NXT runs out of memory

  while (true<>ButtonPressed(BTNCENTER,false)&&(FreeMemory()>=2000)) {
    // ButtonPressed: NXT Guide Section 6.55.2.5 pg. 513
    // BTNCENTER: NXT Guide Section 6.95 pg. 693.  Refers to NXT Orange button
    // returns TRUE if pressed.  Input of "false" means don't reset button count
    // FreeMemory: NXT Guide Section 6.59.2.8 pg. 539.  # of bytes of flash
    // memory available.  Thus loop executes if Orange button not pressed
    // AND there is still more than 2Kbytes left in flash.

    curAngleInDegrees = MotorRotationCount(MOTOR); // get relative position
    deltaAngleInDegrees = curAngleInDegrees - prevAngleInDegrees;
    strDeltaAngleInDegrees = FormatNum("deltaAngle = %ld", deltaAngleInDegrees);

    curTick = CurrentTick(); // read timer value
    deltaT = curTick - prevTick; // measure time elapsed between angle reads
    strDeltaT = FormatNum("deltaT = %ld", deltaT);
    elapsedTimeInSeconds = elapsedTimeInSeconds + (deltaT/1000.0); // in seconds
```

Code Example: Part 4 – the `main` loop for `helloWriteMotorSpeedToFile1_0`

Step 5: `main` continues. The endless while loop calls the NxC statement `CurrentTick()` to set up a timer. `CurrentTick` reports an integer. This integer is the number of milliseconds that has elapsed since the brick was turned on. By calling and storing the value in `CurrentTick()` to variables like `prevTick` and `curTick`, one can calculated elapsed time. Continue typing the following in your program.

```
    // NB: Found bug: need to multiply by a float to force motorRpm to be float
    // If use motorRpm = deltaAngleInDegrees / deltaT * DEG2RPM;
    // then motorRpm will return zero

    motorRpm = deltaAngleInDegrees * DEG2RPM / deltaT;
    strMotorRpm = FormatNum("%5.3f", motorRpm);

    // Display on NXT Brick
    TextOut(0, LCD_LINE2, strDeltaT);
    TextOut(0, LCD_LINE4, strDeltaAngleInDegrees);
    TextOut(0, LCD_LINE6, strMotorRpm);

    prevTick = curTick;
    prevAngleInDegrees = curAngleInDegrees;

    Wait(25);  // update display every 25 milliseconds

    // write elapsed time in seconds and motor speed in RPM to file
    strElapsedTimeInSeconds = FormatNum("%5.3f", elapsedTimeInSeconds);
    text=StrCat(strElapsedTimeInSeconds, ",", strMotorRpm,"," );

    // write the text to a file. The text will be end with a EOL
    WriteToFile(text);
  } // end of while

 // close the file
 StopWriteToFile();

} // end of main
```

Code Example: Part 5 (final part) – the `main` loop concludes for `helloWriteMotorSpeedToFile1_0`

In Part 5, the program calculates the motor velocity (in RPM) but measuring the difference in the angles reported by calling `MotorRotationCount(MOTOR)`. The difference is divided by the elapsed time (`deltaT`). Strings are formatted so that they can be displayed on the brick's LCD. Data is captured to the file. The while loop exits (when say, the brick's orange button is pressed, or the brick's flash memory is maxed) and then calls `StopWriteToFile()` to gracefully close data files and exit the program.

Step 6: Save, compile and download your program to your brick. Attach a motor to Port A. Execute your program. Your motor should move at its maximum speed. The brick will occasionally beep (every time a new data file is created). Hit the orange button after about 5 seconds.

Step 7: With your brick attached to your PC (via USB), click Tools – NXT Explorer (see **Figure 1A**). You should see your data files (as in **Figure 1B**).

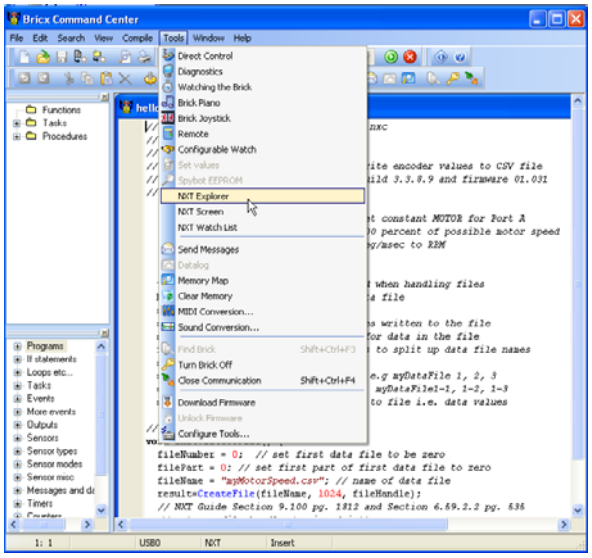

Figure 1A: From menu, Tools – NXT Explorer

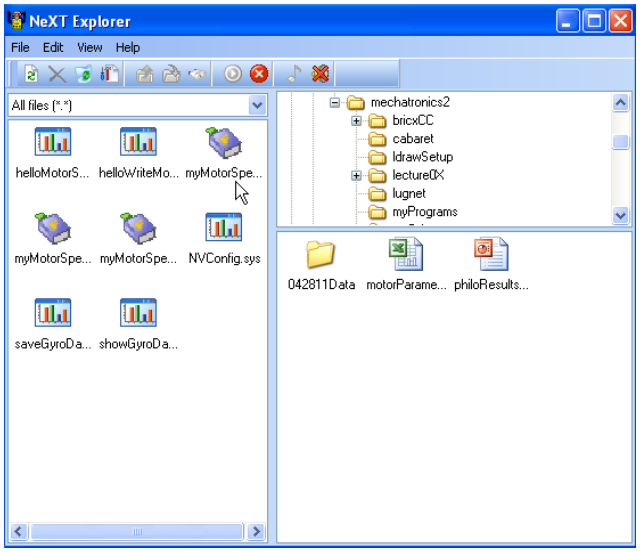

Figure 1B: Contents of NXT Brick (left pane)

Step 8: In NXT Explorer, use the right pane (see Figure 1B) and navigate to a directory and folder (e.g. `myNxtData`) you'd like to import your data files to. For example, this can be a folder on your USB drive. Next, click-and-drag the files you wish to your desired directory's folder (see **Figure 2**)

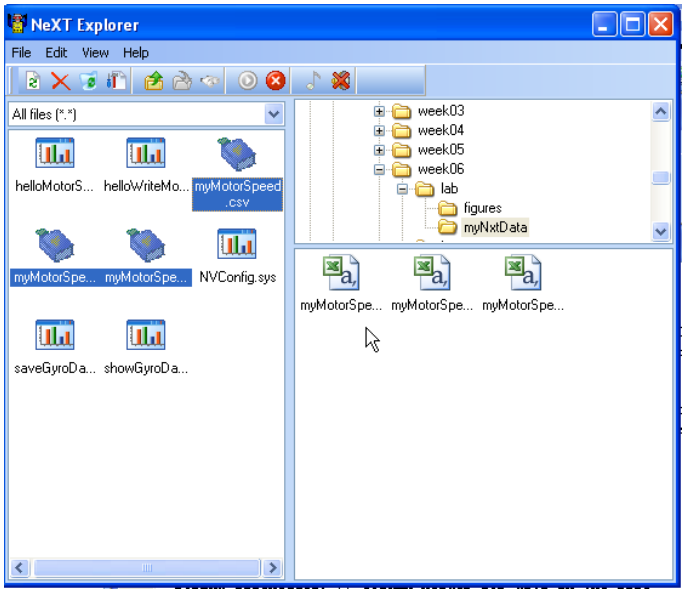

Figure 2: Click-and-drag data files to your PC's folder (e.g. `myNxtData` folder)

Step 9: You can now use Excel to open these data files. Each data file contains 1024 bytes of data. Use Excel to create a single data file, which you will use for plotting. Save this as an Excel file e.g. motorSpeed.xls (see **Figure 3A**). Use Excel Chart Wizard (XY Scatter) to plot the resulting to display the motor speed versus time (see **Figure 3B)**. You should get a result that looks like **Figure 3C**

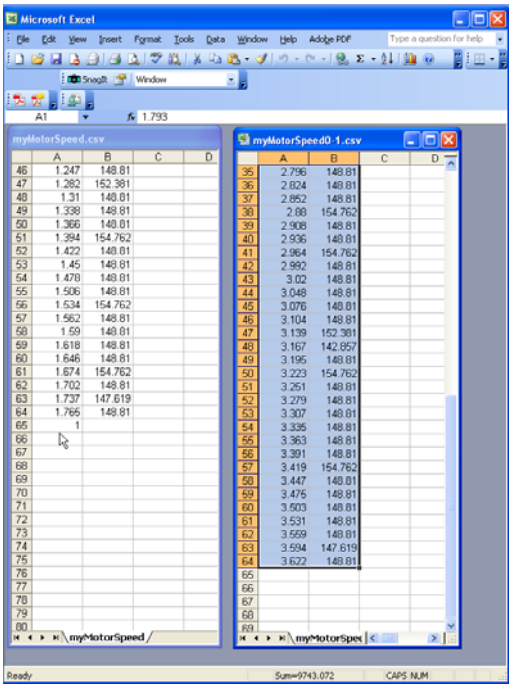

Figure 3A: Open the various .csv files to create and save as a single XLS file.

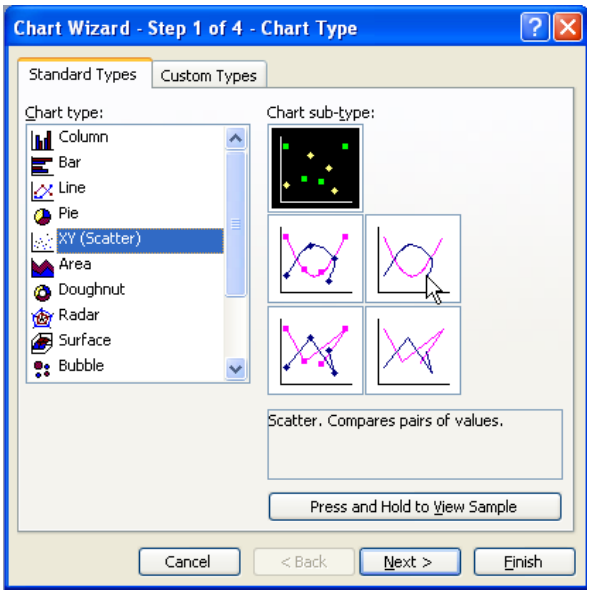

Figure 3B: Highlight the columns you wish to plot and then use Excel's Chart Wizard and XY (Scatter).

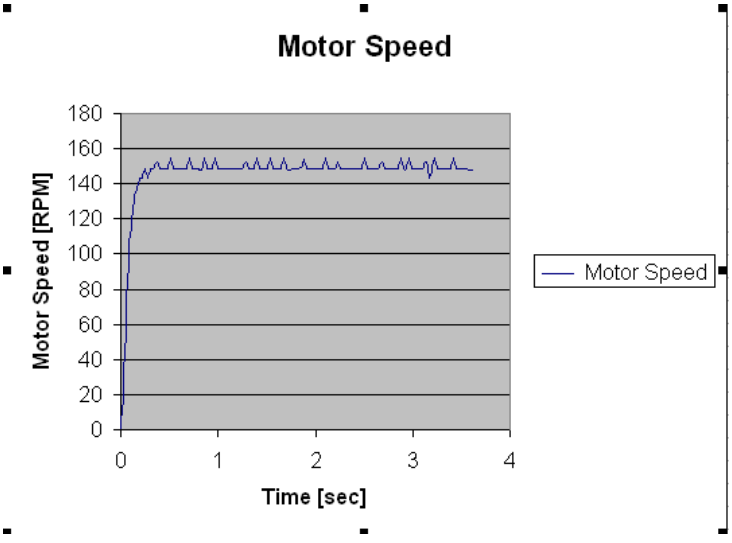

Figure 3C: Resulting Excel plot. Clearly shows that the motor speed is a first-order system. The steady-state speed is about 150 RPM

Exercise 2 In NxC create programs for the following:

2-1 The rise time is defined as the time it takes the system to reach 63% of steady-state. Eyeball your graph (generated as in Figure 3C). What is the rise-time in seconds?

2-2 Command your motor to run at 50% of power. Compile and execute to capture and plot a figure similar to Figure 3C. What is the steady-state velocity? What is the rise-time?