Hands-on Lab

Mindsensors MTRMX-Mx Motor Controller

Mindsensors is a third-party company that provides a wide-range of LEGO-related products. The MTRMX-Nx is a product that can drive up to 4 DC motors. This is important, especially when one wants to use more powerful motors with the NXT. The MTRMX-Nx can handle up to 35 V at 2 Amps.

A motorized propeller will be attached to the damped compound pendulum (DCP). When the motor rotates, the propeller generates lift. The lift creates a torque about a pivot where the DCP is connected to the angle sensor. This lift torque is balanced by the gravitational torque acting on the DCP.

There are no standard LEGO motors that can rotate a propeller fast enough to generate lift. The running current can also be quite high (2 A), depending on the DCP's lever-arm length. As such, a non-LEGO 5V DC motor with an external 9 V, 2 Amp power supply are used.

Step 1: Hook up your MTRMX-Nx with the following connections shown in **Fig 1-1A**.

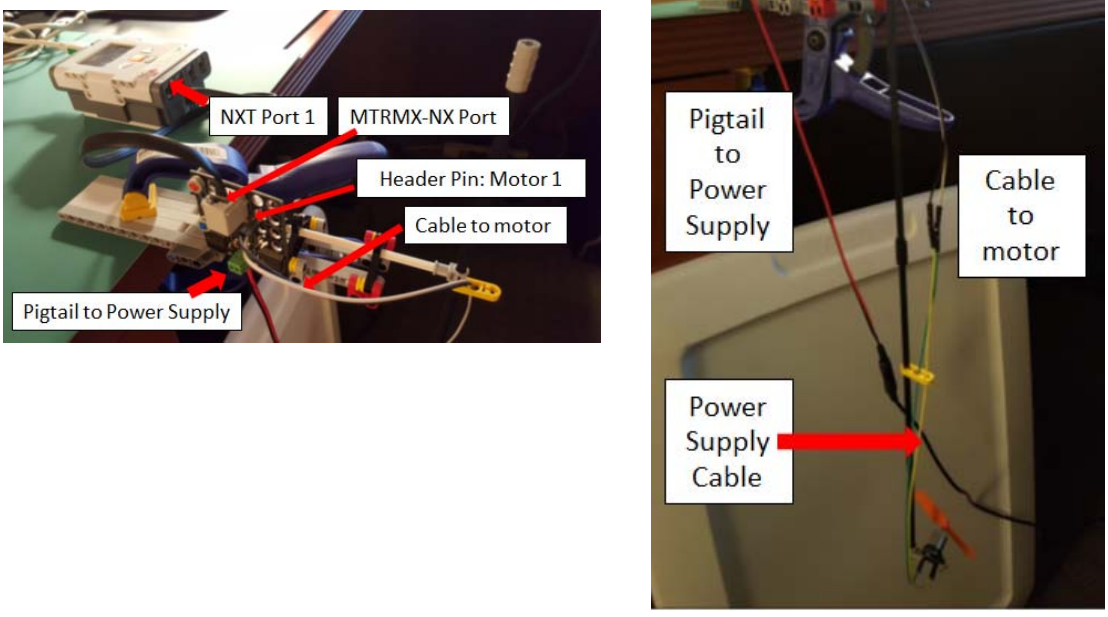

Fig 1-1A: Wiring connections shown in left photo: (1) Connect Lego cable between MTRMX-Nx port and NXT brick port 1; (2) Connect pigtail to MTRMX-Nx green screw terminal where red wire goes to +'ve and black wire goes to –'ve positions of connector; and (3) Connect motor cable to MTRMX-Nx 2-pin header (Motor 1). Right photo shows pigtail connects to the wall-wart style power supply and the motor cable connects to the motor's leads.

Step 2: Enter and compile the following code: `helloMtrmx-Nx1_1.nxc`

```
// FILE: helloMtrmx-Nx1_1.nxc
// VERS: 1.0: sends speed of 128 to DC motor - works!
//       1.1: motor speed changes rotation based on arrow push - works!
// DESC: DC motor connected to Mindsensors MTRMX-Nx board
//       MTRMX-Nx connects to Brick Port 1
//       DC motor connects to Header Pin 1 on MTRMX-Nx
// REFS: Mindstorms demo program MTRMUX-demo.nxc

const byte motorPort  =  IN_1;   // Brick Input Port 1
#define ADDR     0xB4            // I2C address: Factory set to 0xB4
#define MOTOR_1  1               // DC motor connected to 2-pin header

// Mindsensors MTRMX-Nx manual defines following for command registers
#define MC_FLOAT    0x00
#define MC_FORWARD  0x01
#define MC_REVERSE  0x02
#define MC_BRAKE    0x03

void MS_MTRMXControl(byte port, byte i2cAddr, byte motorNumber, byte
commandDirection, byte commandSpeed)
{
  byte location;
  byte message[20];
  byte nByteReady = 0;

  SetSensorLowspeed(port);

  location = 0x40 + motorNumber*2; // i.e. 0x43 sets Motor 1's speed

  // NxC Language: ArrayBuild constructs single array "message" from elements
  ArrayBuild(message, i2cAddr, location, commandDirection, commandSpeed);

  // ping I2C port to see if ready to receive message
  while (I2CStatus(port, nByteReady) ==  STAT_COMM_PENDING);
  // I2C port ready to receive a message
  // write message to port.  Note: no return data is expected, hence 0
  I2CWrite(port, 0, message);
  // ping I2C port to see of message completed
  while (I2CStatus(port, nByteReady) ==  STAT_COMM_PENDING);
}

task main() {

  byte motorSpeed;     // motor speed from 0 to 255
  byte motorDirection; // either MC_FORWARD (0x01) or MC_REVERSE (0x02)

  bool orangeButtonPushed, leftArrowButtonPushed, rightArrowButtonPushed;

  TextOut (0, LCD_LINE1, "Arrow: CW/CCW" );
  TextOut (0, LCD_LINE2, "Orange quits" );

  // NxC language: configures an I2C sensor i.e. MTRMX-Nx connected
  // to IN_1 (Brick's Input Port 1)
  SetSensorLowspeed(motorPort);
  motorSpeed = 128; // set motor speed to 128 (range is 0 to 255)
  motorDirection = 0; // set default direction for CW rotation
```

```
  do {
   leftArrowButtonPushed = ButtonPressed(BTNLEFT, FALSE);
   rightArrowButtonPushed = ButtonPressed(BTNRIGHT, FALSE);
   orangeButtonPushed = ButtonPressed(BTNCENTER, FALSE);

   if(rightArrowButtonPushed) motorDirection = MC_FORWARD; // CW direction
   if(leftArrowButtonPushed) motorDirection = MC_REVERSE;  // CCW direction

   // Start Motor 1
   MS_MTRMXControl(motorPort, ADDR, MOTOR_1, motorDirection, motorSpeed);
   Wait(100);
  } while(!orangeButtonPushed);

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

  ClearScreen();
  TextOut(0, LCD_LINE2, "Quitting", false);
  motorSpeed = 0;
  MS_MTRMXControl(motorPort, ADDR, MOTOR_1, motorDirection, motorSpeed);
  PlaySound(SOUND_LOW_BEEP);  // Beep to signal quitting
  Wait(SEC_2);

} // end of main
```

Code Description:

Beginning in main, the code `helloMtrmx-Nx1_1.nxc` defines variables `motorSpeed` and `motorDirection` as well as Boolean variables to detect NXT button states.

The MTRMX-Nx uses I2C to connect to the brick. As such, the MTRMX-Nx calls the NXC function `SetSensorLowSpeed(motorPort).` `motorPort` is the NXT port (1, 2, or 3) where the MTRMX-Nx is connected to. In the above program, we are using Port 1.

Next, the program assigns byte values (numbers ranging from 0 to 255) to `motorSpeed` and `motorDirection`. The program then checks NXT button states to assess if the user wants to start the motor or quit.

Mindsensors provides a function called:

```
void MS_MTRMXControl(byte port, byte i2cAddr, byte motorNumber,
byte commandDirection, byte commandSpeed)
```

As such, to command the motor to move, the above program has the statement:

```
MS_MTRMXControl(motorPort, ADDR, MOTOR_1, motorDirection,
motorSpeed);
```

`motorSpeed` can have values ranging from `0` to `255` (motionless to full speed). `motorDirection` can have values of `0` or `1` (clockwise or counter-wise motor rotation).

The do-while loop continues to monitor NXT button states. When the user hits the orange button, the loop is exited, the motor commanded to stop, and a beep is played.

Exercise: Write and demonstrate the following program:

1-1: Set motor direction to 0. Starting at zero, increase motor speed by 25 each time the user hits the right-arrow button. Display the motor speed on the NXT LCD's display. Your code should not exceed a motor speed command of 250.

1-2. Set motor direction to 0. Starting at 255, decrease motor speed by 25 each time the user hits the left-arrow button. Display the motor speed on the NXT LCD's display. Your code should not exceed a motor speed command of 5.