.NET nanoFramework

And Lego hacking

Peter Shaw

Jose Simões

Tonight’s Session

- Introduction to .NET nanoFramework – Jose Simões
- .NET nanoFramework compatible devices – Jose Simões
- nanoFramework getting started demo – Peter Shaw
- Hacking Lego NXT (How to) – Peter Shaw
- Creating an I2C slave device for NXT – Jose Simões
- Lego NXT Motors Demo – Peter Shaw

About Me

- Had a healthy obsession with dismantling all manner of electronical devices (Software and Hardware) since I was about 3 yrs old.
- Been doing computing & electronics in one form or another since 1979.
- Own my own technology consultancy called “Digital Solutions” based in the North East UK, which lets people hire me to destroy their gadgets and systems to figure out why they don’t work correctly.
- Been a .NET dev since it’s inception in the early 2000’s and have worked on many beta test teams for .NET related frameworks over the years.

About Jose

- Jose’s about me part goes here.

Introduction to .NET nanoFramework

- Joses slides part introducing nano and showing compatible devices.

Hacking Lego NXT (How to)

- Why Lego NXT?
  - The NXT system is based around a 9V Arm cortex CPU, but the controller brick is quite closed off.
  - The challenge of working out how and why things work the way they do.
  - Industry standard protocols and architectures, once you get under the hood.
- What can we do using Lego NXT
  - ANYTHING that the NXT control brick can do.
  - New sensors for our NXT bricks
  - Reading existing Lego NXT sensors
  - Controlling Lego NXT motors

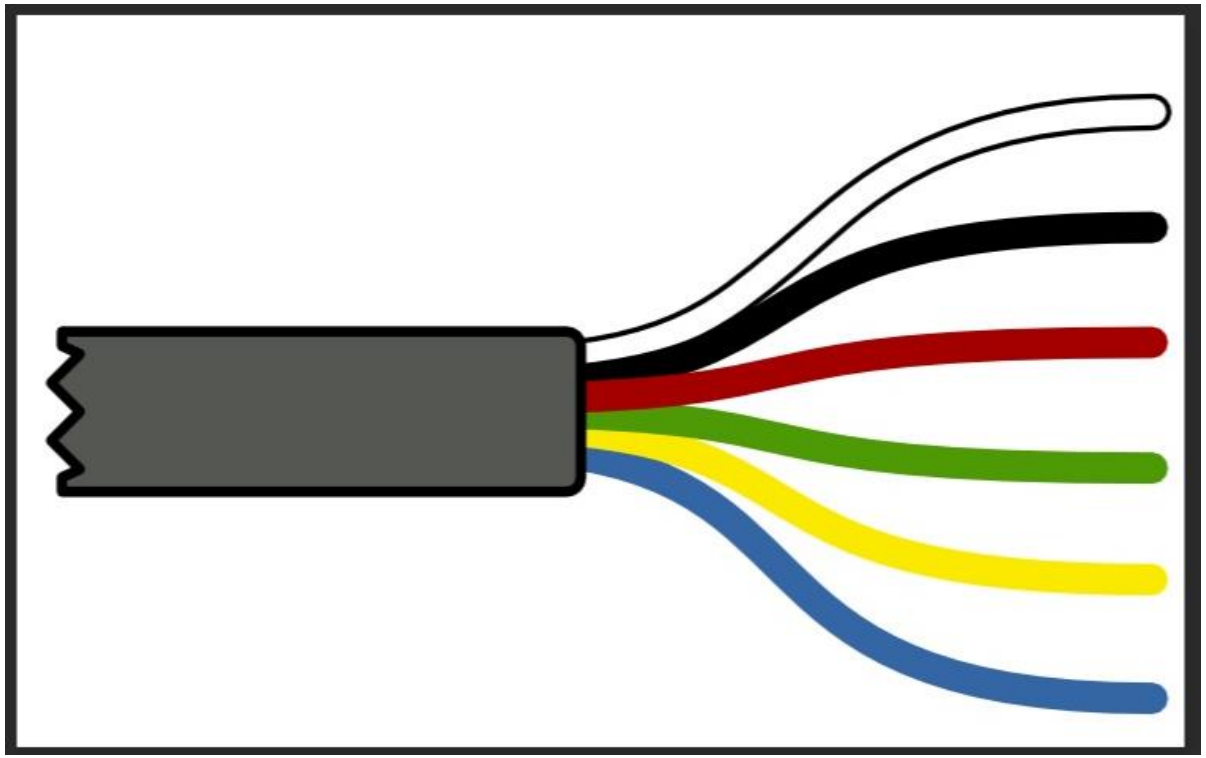
Hacking Lego NXT (How to)

- What can we do using Lego NXT
  - NOT Just NXT, but we can also use EV3, Spike Prime and even the older RCX system components too.
  - We can freely mix Lego components from all 3 revisions of Mindstorms as well as our own creations.

- What modifications do we need to make to our NXT system?
  - The only thing you need to “break” are the connecting cables.
  - Dexter Industries used to do a series of breakout boards for NXT but no longer make them.
  - You can use an RJ12 plug with the clip cut off and glued back on to the right side of the plug.

Hacking Lego NXT (How to)

- What modifications do we need to make to our NXT system?



The easiest method is to CUT Open a spare cable.

Wires are DUAL PURPOSE in most cases depending What you are trying to achieve.

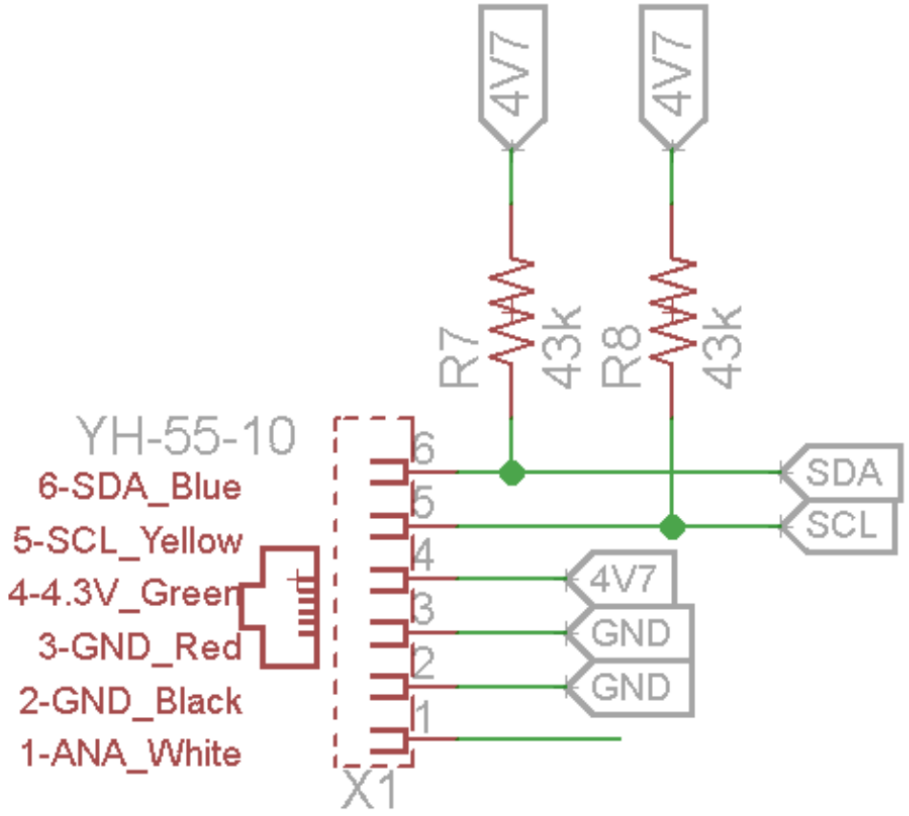
For driving the motors: White/Black are your PWM And ground signals. (These are swapped for direction Control)

For Digital Comms: Yellow is SCL line and Blue is the SDA line on the I2C bus.

Green is 5v power for devices powered by the NXT Red is the system Ground for devices and sensors.

Hacking Lego NXT (How to)

- What modifications do we need to make to our NXT system?



The easiest method is to CUT Open a spare cable.

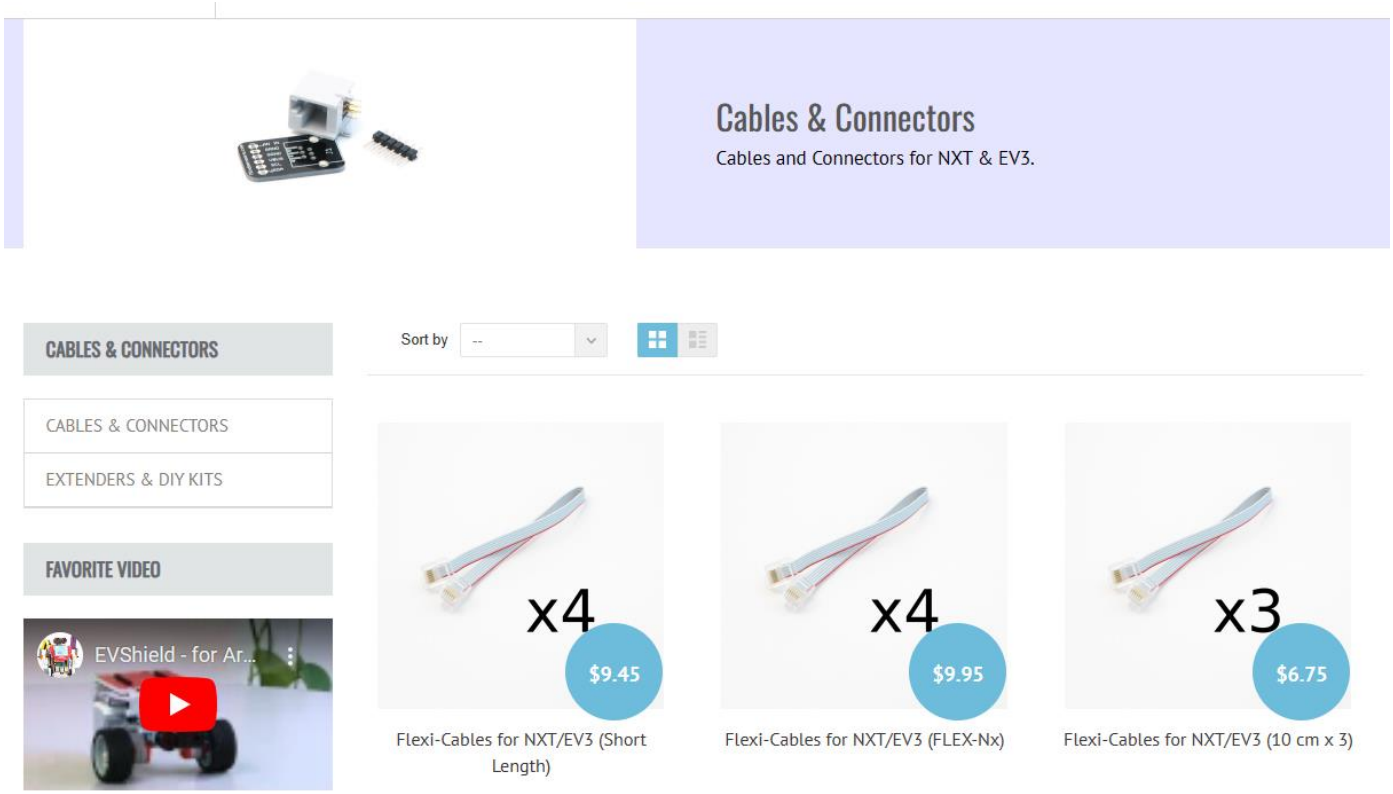
If your looking to produce your own I2C sensors
There are 2 important points to note.

1) You WILL need to terminate the I2C bus with a pair Of 43k resistors as shown in the diagram.

2) The I2C device specification (Particularly with ID Numbers) is not 100% standard, Lego sensors all use A common ID and a sub-ID that the NXT brick Understands, you'll need the NXT hardware manual To understand how this works.

Hacking Lego NXT (How to)

- What modifications do we need to make to our NXT system?



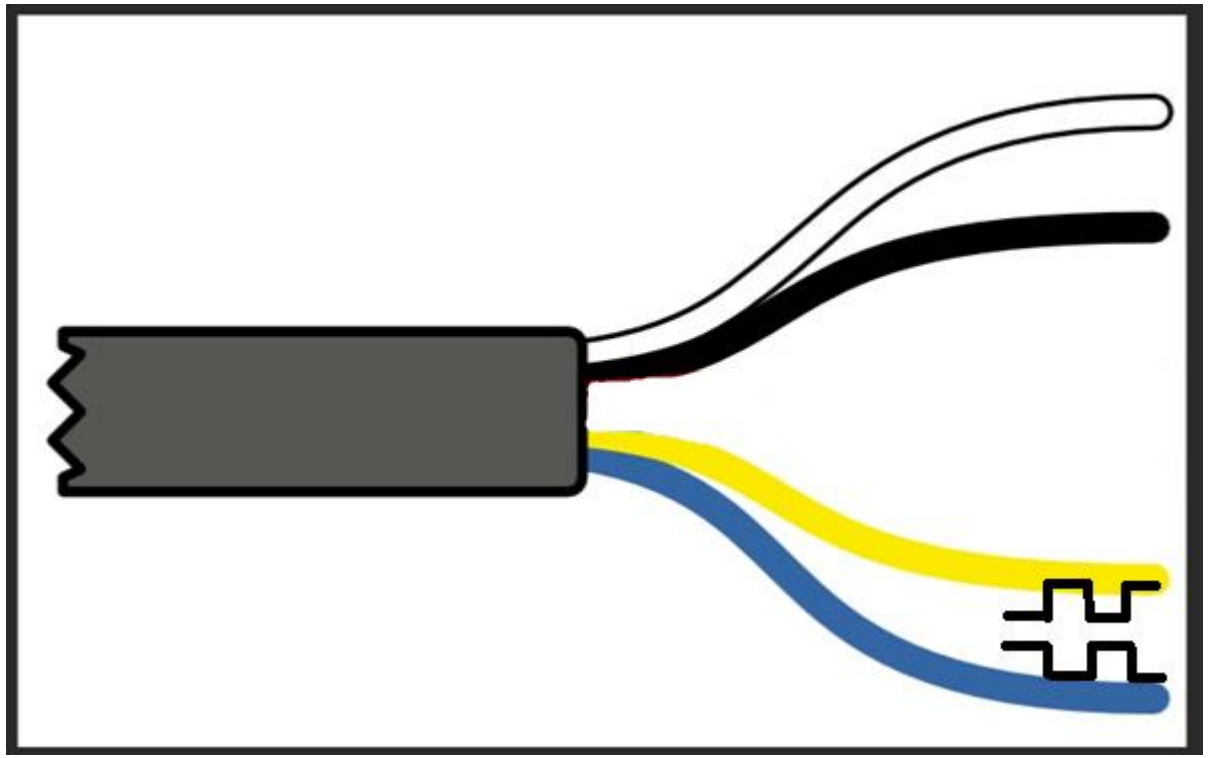
You CAN buy cables from “mindsensors.com” but They can be very pricy depending on length and Quantity required.

I find that it’s much cheaper to buy spare cables From places like eBay, then cut them open so that You keep your original cables intact.

Mindsensors do also sell the plugs and sockets seperately but as they are RJ12 plugs you will need a suitable crimping tool to attach them (AN RJ45 Network Crimp tool WILL NOT WORK!)

Hacking Lego NXT (How to)

- What modifications do we need to make to our NXT system?



If you are only driving motors, and NOT using I2C
On the same cable connection, you can use the Blue
And Yellow digital wires to read the Tacho sensors
Built into the motors.

These 2 lines will produce 2 square waves, each 90 degrees different to each other. Yellow will lead if the motor is being moved forward, blue will lead if the motor is moving in reverse.

Hacking Lego NXT (How to)

- What modifications do we need to make to our NXT system?

If you have a sharp enough knife and a steady hand, it's possible to also take a regular RJ12 telephone (The same type used on American 6 pin connectors).

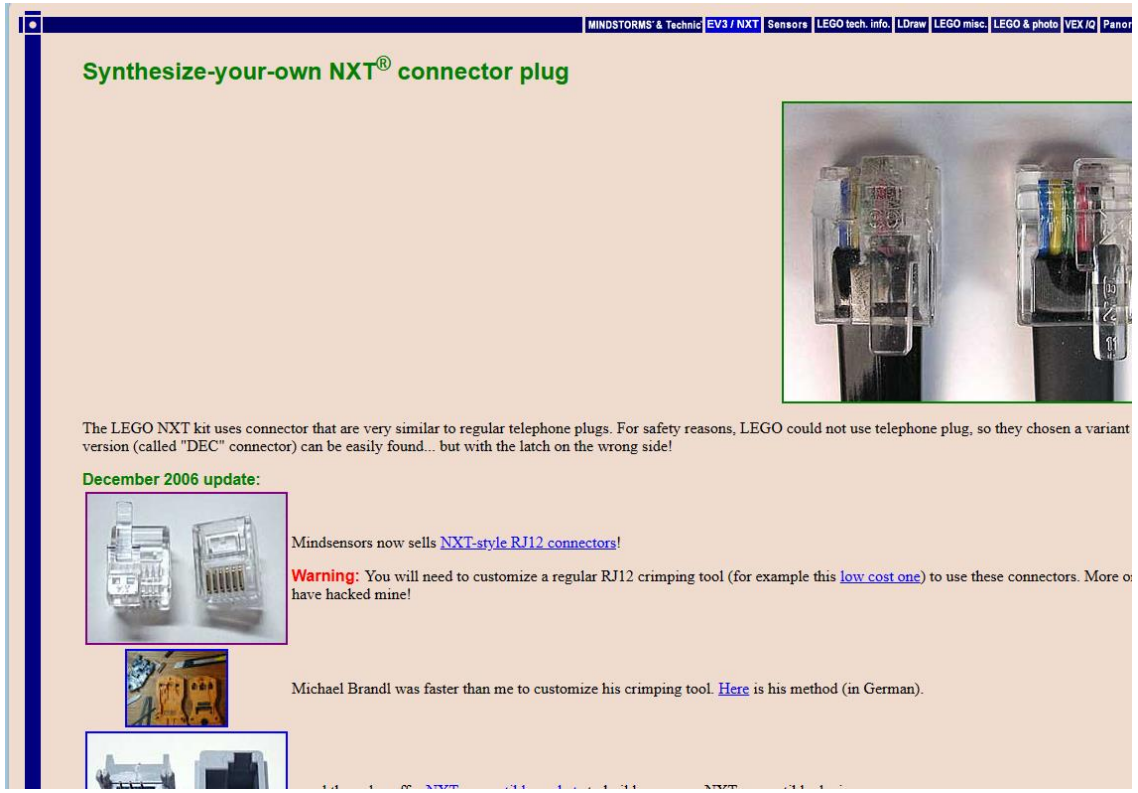
Using the knife, slice under and through the plastic clip, then once the clip is free, move it to the right side of the plug (Looking from the back) and apply some thin super glue to fix it back down.

Mindsensors do sell these plugs, and as far as I know they are the only place that do. There are commercially available plugs but they have the clip on the wrong side, so check carefully when ordering.

Do note that a regular RJ12 crimp tool WILL NOT FIT The modified plugs.

Hacking Lego NXT (How to)

- What modifications do we need to make to our NXT system?



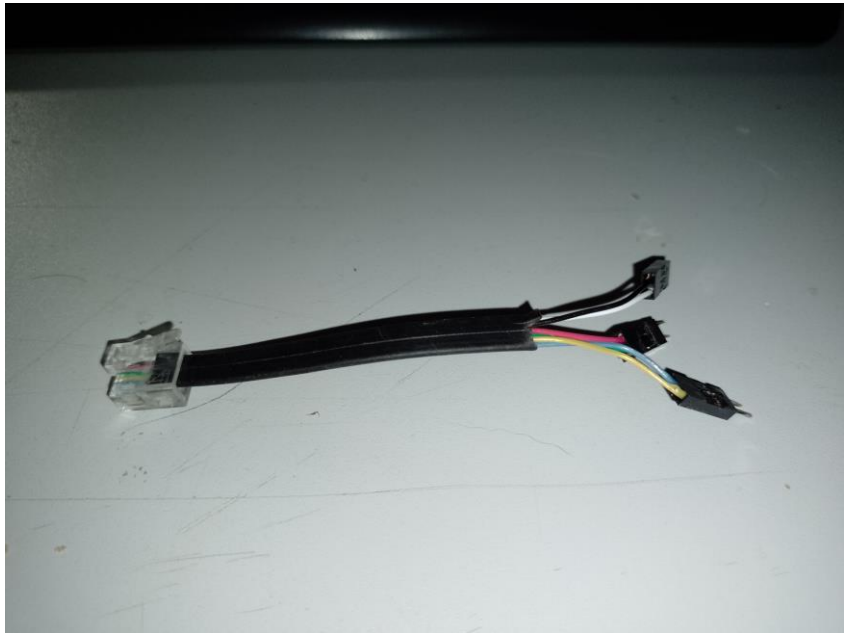
If you do want to make your own plugs, then this Page:

https://www.philohome.com/nxtplug/nxtplug.htm

Contains a huge amount of valuable information Including how to adapt a regular crimping tool, to Fit the modified plugs.

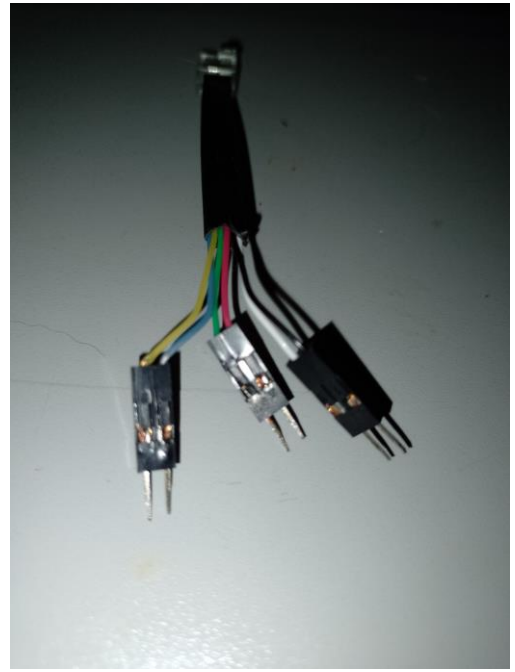
Hacking Lego NXT (How to)

- What modifications do we need to make to our NXT system?



One spare cable and a set of Dupont crimps
Gets you 2 connectors, as you can see from
The photos here I've grouped the 6 pins, into
3 groups of 2.

1 for the motor control
1 for the power
1 for the digital I/O



Hacking Lego NXT (How to)

- What modifications do we need to make to our NXT system?

If you want to connect the older RCX gear
Making a cable is even easier.

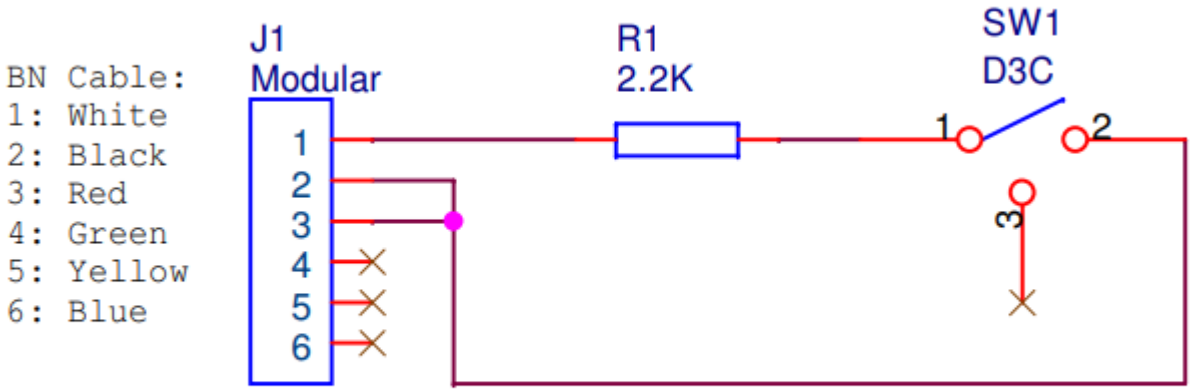
You only have 2 wires.

Each of these 2 wires are used interchangeably
In place of the “pairs” in the NXT system.

So you can use them as a black/white pair for
Motor control, red/green for power or simple
Analog sensor OR blue/yellow for Digital I/O
But never all at the same time as you can for NXT

Hacking Lego NXT (How to)

- Connecting a simple sensor

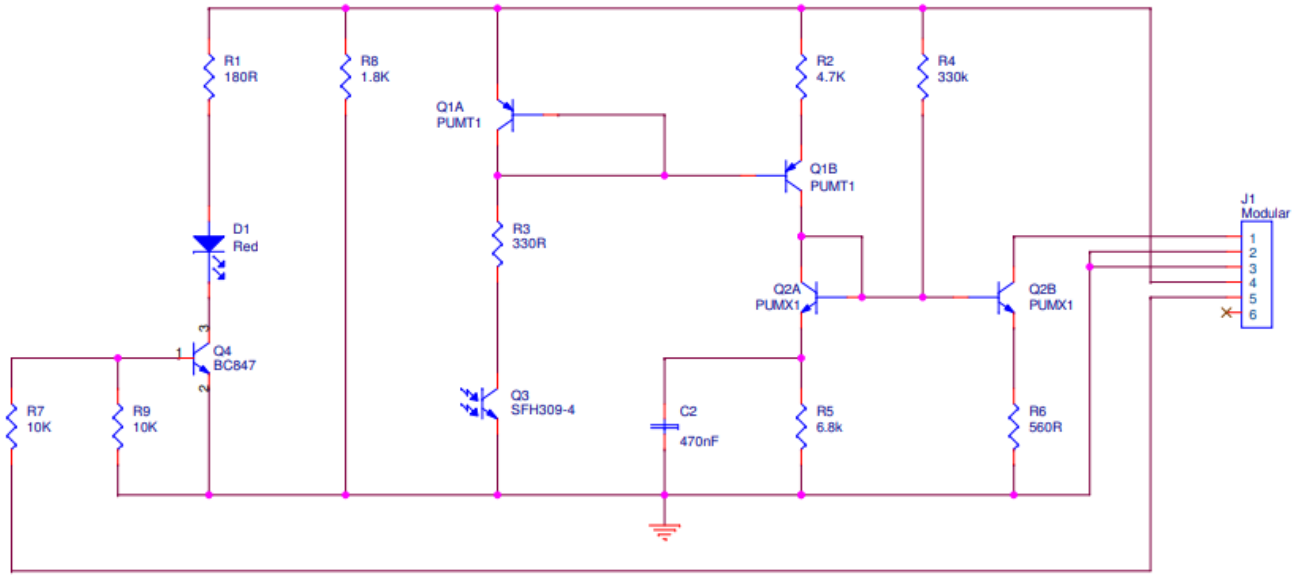


For simple sensors, such as the touch sensor, you simply need to check for A connection across the Black/red and white wires.

we simply tie the motor ground and power ground (Black/Red) together then provide a switched connection to the White wire. This switch could be a push button, or a relay as long as the connection is 2 state the NXT Will see it as a Lego touch sensor, in reverse, we can Simply use the GPIO framework in .NET nano to Check a pin connection from white to black/red in The usual manner that you would Normally read a pin on your MCU.

Hacking Lego NXT (How to)

- Connecting the NOT so simple sensors



For the other sensors, it depends entirely on which sensor your trying to use.

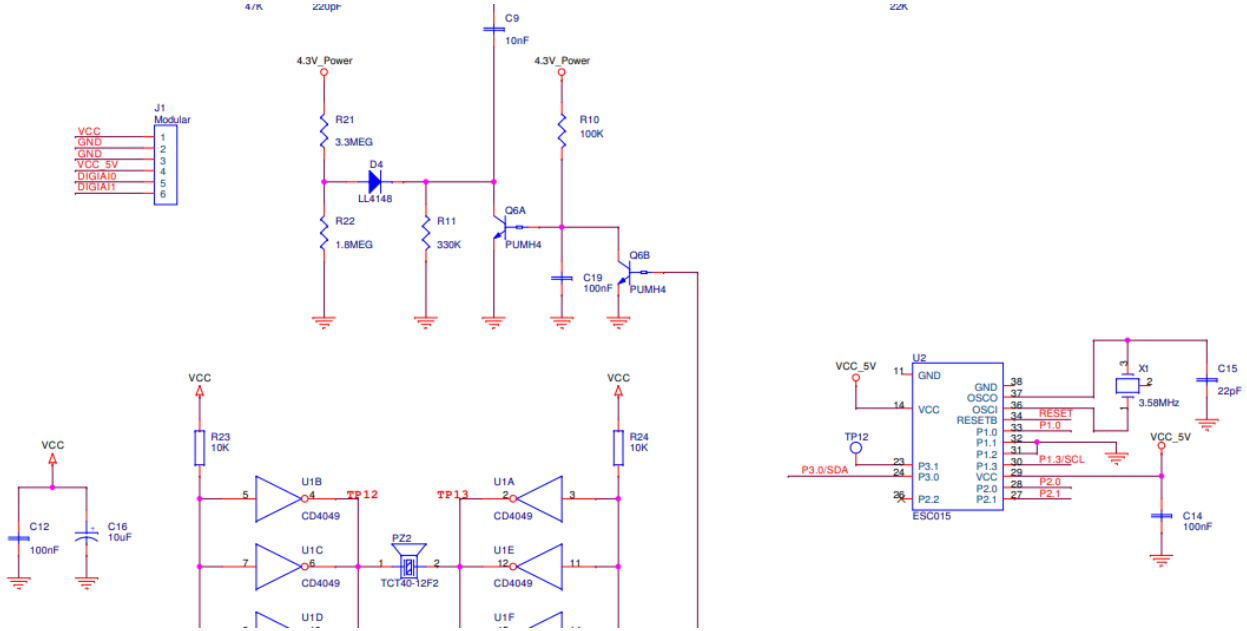
The light sensor for example, shown here doesn't Provide any digital I/O signals such as an I2C, but it Does repurpose the SCL line as well as the power And PWM lines to provide differential signals showing The relative strengths of the intensity's it can see Under the sensor.

This requires you to look for pulses on the SCL line While measuring analogue voltages on the PWM lines Used by the motors.

As mentioned previously wires are dual purpose in Many cases.

Hacking Lego NXT (How to)

- Connecting an I2C sensor



Likewise, connecting the I2C sensors is not USUALLY Just as simple as connecting up to the I2C bus and Reading, you do need to provide the power and in the case of the ultrasonic sensor the trigger to start A read for it.

.NET nano however, does now have a very good set Of methods for creating I2C slave devices, so at this Point I'm going to hand back to Jose who will take You through some I2C slave sample code and Explain how .NET nano can be used to create a Device that could be interfaced to your NXT controller.

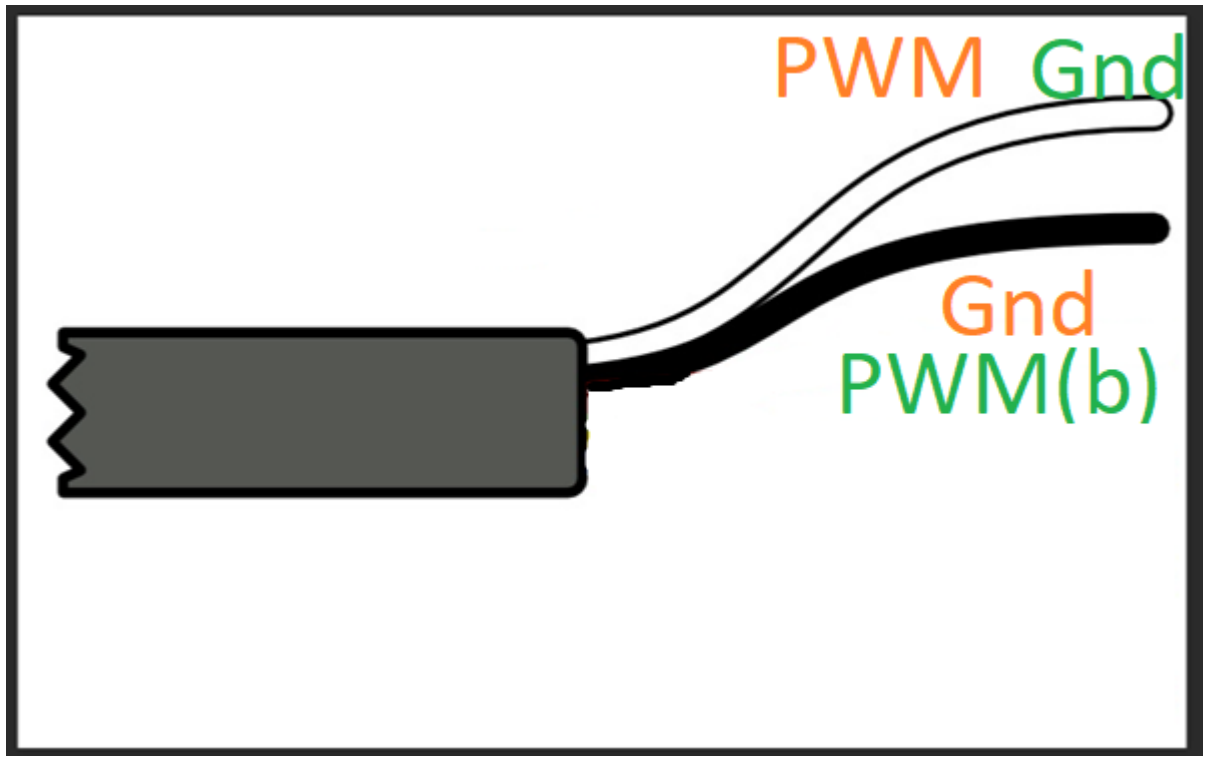
I will be making a download of the Lego NXT circuit Diagrams available after this session is finished, please Keep an eye out for the details.

Jose’s I2C slave part

- Jose’s I2C slave code explanation goes here.

- Remember to explain and introduce before handing over….

Driving an NXT Motor using .NET nano



To drive an NXT motor we drive a 9V PWM signal onto Either the white or the black wires, while grounding the Opposite wire in the pair.

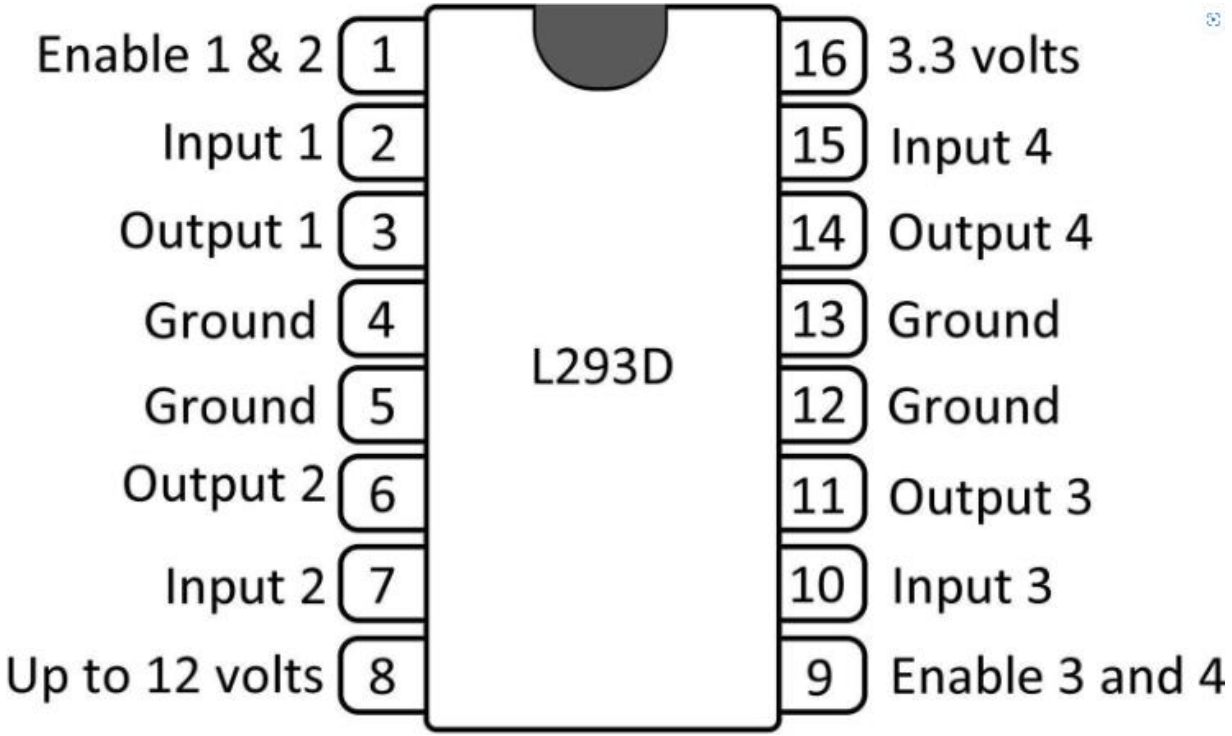
The duty cycle of the PWM sets the speed of the motor 100% motor is at full speed, 50% half speed and so on.

We apply PWM to white and Ground to Black for forward motion and PWM to Black and Gnd to White For backward motion.

While you can use your own arrangement of transistors And/or BJT's it's much easier to use a dedicated motor Driver IC such as the L293D driver chip or one of its More up to date alternative's.

DO NOT HOWEVER TRY TO DRIVE THE MOTOR DIRECTLY OFF YOUR MCU!!! BAD THINGS WILL HAPPEN.

Driving an NXT Motor using .NET nano



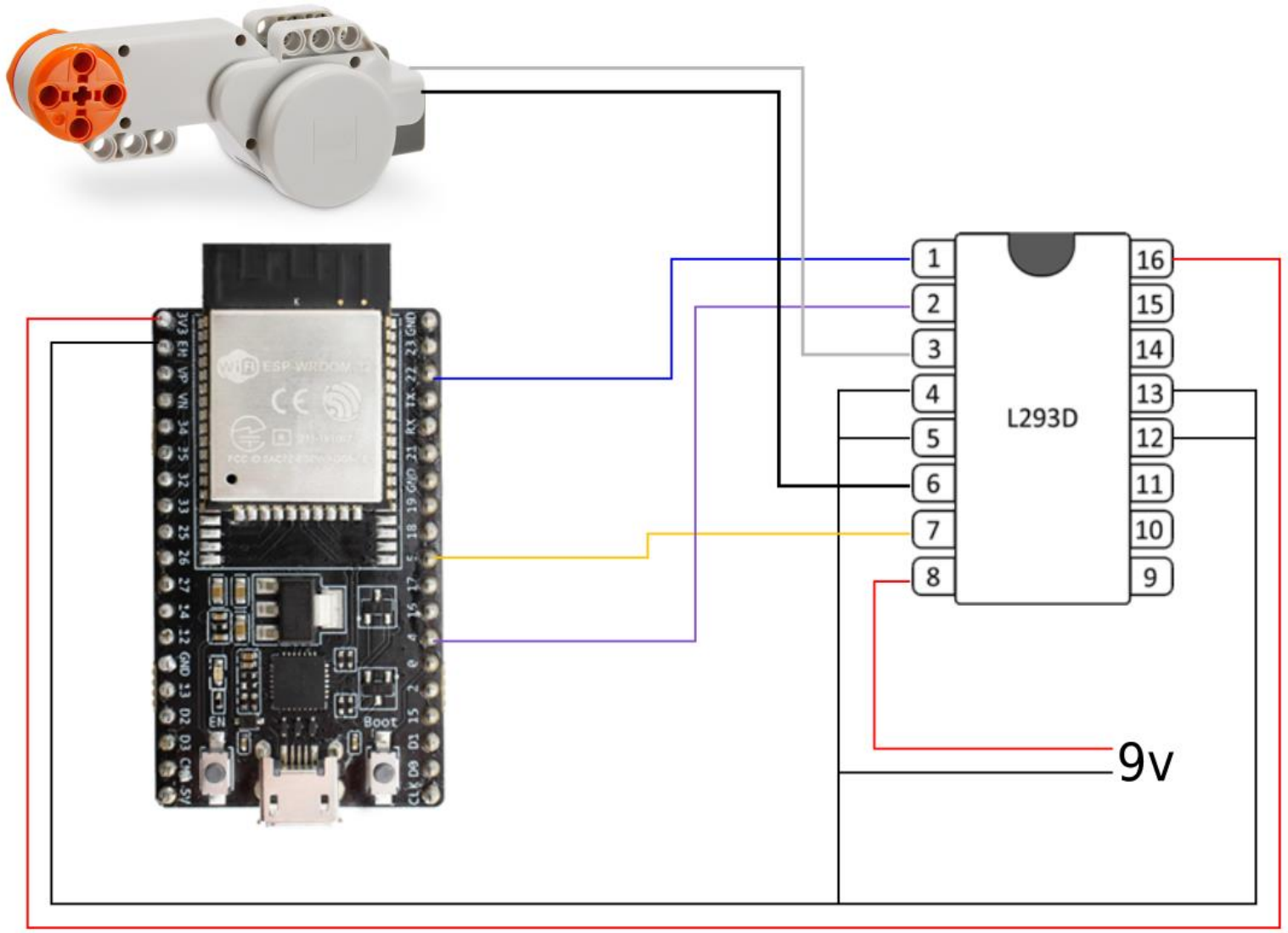
The L239D can drive 2 motors at the same time, for Lego NXT however, the motors draw just a little bit More current than a normal DC motor usually does due to the gearing they have built into them, so it's best to just use one IC for each one motor you want to drive.

You'll need a separate 9V supply (Applied on pin 8) to Power the motors themselves, everything else can be Obtained from the 3.3v rail on the MCU (In this case an ESP32 Devkit2).

From the MCU itself (and controlled by .NET nano) you Will need one PWM pin which is connected to pin 1 on The motor driver, and 2 general purpose digital GPIO Pins that are connected on pins 2 & 7 on the IC.

The black/white PWM pair on the NXT motor connect To pins 3 & 6 on the driver IC.

Driving an NXT Motor using .NET nano



Points to Note:

1) You MUST connect all your grounds to the MCU ground.

2) Keep the 9v away from your other voltage lines.

3) You MUST tie all 4 grounds on the motor driver together.

4) If you use a heat sink on the driver, make sure it’s grounded.

Driving an NXT Motor using .NET nano

```
12      private const int Input1Pin = 4;
13      private const int Input2Pin = 5;
14      private const int MotorEnablePin = 22;
15
16      private static GpioPin Input1;
17      private static GpioPin Input2;
18      private static PwmChannel MotorEnable;
```

We start our .NET nano code by defining a few Constants, then creating our app variables.

We have constants for our PWM pin, and our GPIO pins used to control the motor.

(I'm showing the code as screen shots here simply Because in it's entirety, it's less than 100 lines long And a great chunk of that is whitespace and code that's For show rather than being required)

Driving an NXT Motor using .NET nano

Next up we create our “Main” function as is common
With all .NET nano applications, and then we configure
Our PWM pin.

We set our PWM frequency to 40Hz and attach it to our
“MotorEnable” variable using the constant we
Defined for the pin number.

```
0 references
public static void Main()
{
    Configuration.SetPinFunction(MotorEnablePin, DeviceFunction.PWM1);
    MotorEnable = PwmChannel.CreateFromPin(MotorEnablePin, 40000, 0);
```

Driving an NXT Motor using .NET nano

Lastly we use a GPIO controller from the nanoFramework libraries, set our Input1 & 2 variables to be Appropriate place holders for those pins, numbering them using our pin number constants, and setting The mode on them to output.

PLEASE do note the comment! I've not tried it, but reading the motor driver data sheet, states that it will attempt To drive BOTH PWM lines to the motor to a positive PWM voltage at the same time, something which I cannot Imagine that anything good would be the result. Other motor drivers may differ.

```
25          var gpioController = new GpioController();
26
27          Input1 = gpioController.OpenPin(Input1Pin, PinMode.Output);
28          Input2 = gpioController.OpenPin(Input2Pin, PinMode.Output);
29
30          // NEVER SET BOTH INPUTS HIGH AT THE SAME TIME!
```

Driving an NXT Motor using .NET nano

To complete our initialisation, we turn BOTH outputs off so nothing is driving the motor, and we set our PWM Duty cycle to 50%.

I've found while doing this that a duty cycle of less than 0.5 on these NXT motors results in no drive at all, even Though using an oscilloscope I can see a valid PWM signal, I don't know if it's the motor design, the gearing or Anything else, the 50% speed does match the slowest speed you can get from an NXT brick however so I suspect It's by design.

```
31
32          // Set the motor enable pin duty cycle
33          MotorEnable.DutyCycle = 0.50;
34
35          // Motor Off
36          Input1.Write(PinValue.Low);
37          Input2.Write(PinValue.Low);
38          Thread.Sleep(1000);
```

Driving an NXT Motor using .NET nano

From this point on, we simply set input 1 high while 2 is low to drive the motor forward, and 2 high while 1 is low To drive the motor backwards.

As mentioned before do not set both inputs high at the same time, when changing direction it's better to switch the high OFF first before switching the opposite pin on, effectively always writing 0,0 to both pins before then Writing the direction you want.

```
40              // Motor Forward
41              Input1.Write(PinValue.High);
42              Input2.Write(PinValue.Low);
43              Thread.Sleep(1000);
```

Driving an NXT Motor using .NET nano

```
40          // Motor Forward
41          Input1.Write(PinValue.High);
42          Input2.Write(PinValue.Low);
43          Thread.Sleep(1000);
44
45          MotorEnable.DutyCycle = 0.50;
46          MotorEnable.DutyCycle = 0.55;
47          MotorEnable.DutyCycle = 0.60;
48          MotorEnable.DutyCycle = 0.65;
49          MotorEnable.DutyCycle = 0.70;
50          MotorEnable.DutyCycle = 0.75;
51          MotorEnable.DutyCycle = 0.80;
52          MotorEnable.DutyCycle = 0.85;
53          MotorEnable.DutyCycle = 0.90;
54          MotorEnable.DutyCycle = 0.95;
55          MotorEnable.DutyCycle = 1;
56          MotorEnable.DutyCycle = 0.95;
57          MotorEnable.DutyCycle = 0.90;
58          MotorEnable.DutyCycle = 0.85;
59          MotorEnable.DutyCycle = 0.80;
60          MotorEnable.DutyCycle = 0.75;
61          MotorEnable.DutyCycle = 0.70;
62          MotorEnable.DutyCycle = 0.65;
63          MotorEnable.DutyCycle = 0.60;
64          MotorEnable.DutyCycle = 0.55;
65          MotorEnable.DutyCycle = 0.55;
66          MotorEnable.DutyCycle = 0.50;
```

You can however change the PWM duty cycle while
The motor is running, thus changing the motor speed
Without stopping the motor first.

Wrapping all this up in a small library is not difficult
And at some point I will probably most likely create a
Library to add to the ever growing “official” drivers
Repository for .NET nano which already has a
Staggering amount of support for many of the hobbyist
Boards and electronics components that are generally
Available.

Physical Motor Demo Running

- Do motor demo here