

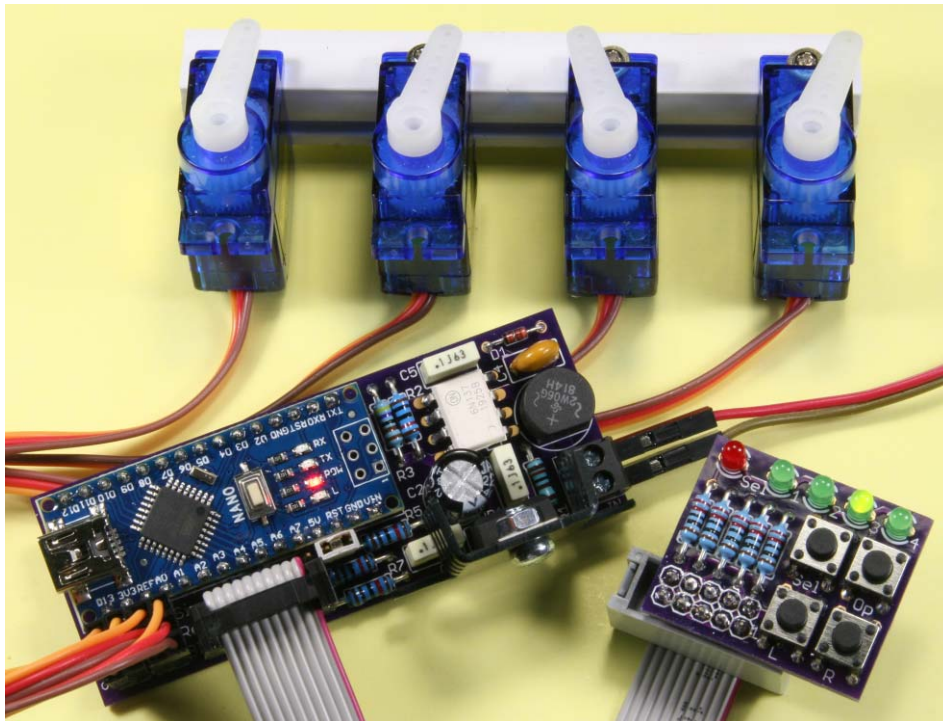
Build Your Own Quad Servo DCC Decoder – Part 1

Introduction

After deciding to use servos to drive the turnouts on a small HO layout which I am planning to build, trials with a number of commercial DCC accessory decoders were rather disappointing. I wanted to be able to set up the throw of each servo easily and accurately by hand, prior to their installation on the layout, and then be able to fine-tune the servo movements after they were in place.

I found the facilities offered by the commercial decoders rather awkward to use, especially once the servos and decoder were in place under the layout, and it seemed quite difficult to set the servo throws accurately. Hence, inspired by the work of Geoff Bunza on accessory decoders based around Arduino modules, I decided to build my own, with the capability of setting up and operating four servos. The successful end result is a DCC accessory decoder which can be built by anyone with reasonable soldering skills and the basic computer knowledge to load software into an Arduino module, for a cost of less than \$20, including custom-designed printed circuit boards.

The Quad Servo DCC Decoder (QSDD) actually consists of two separate modules, the decoder itself which connects to the layout track and drives up to four attached servos –



All photographs by the author

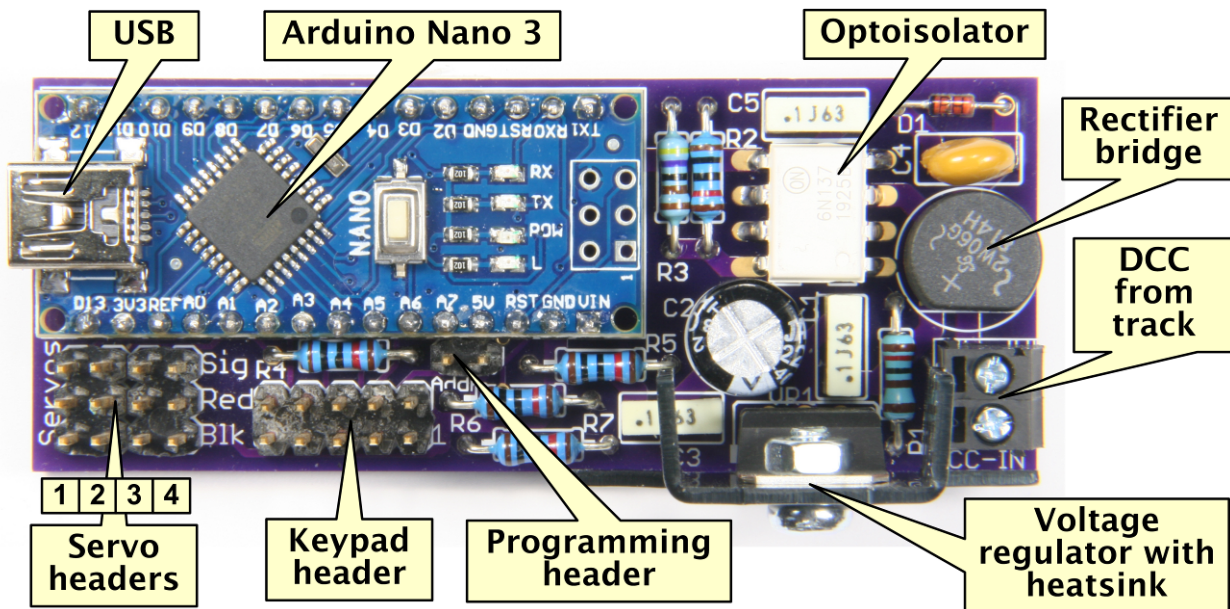
- and a small detachable keypad which plugs on to the decoder and is used to manually adjust the servo throws and set the speed of servo movement.

The keypad can also be used to operate the servos by hand, using the fitted pushbuttons. For convenience, as well as being able to be plugged directly into the decoder, the keypad can alternatively be connected via a ribbon cable up to 40 inches long, allowing you to see and set up your turnouts when the decoder and servos are out of sight under the layout.

Note that, once set up, there is no need to keep the keypad plugged in. It can, if you wish, be completely disconnected from the decoder board. The decoder will then drive the connected servos to operate your turnouts in response to standard DCC commands sent to the track from your command station, using either handheld controllers or suitable software (such as [JMRI Panel Pro](#) or my own [A-Track](#) application) running on your computer.

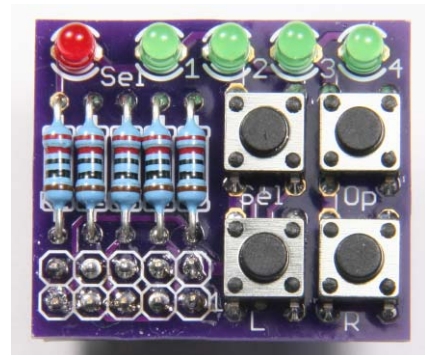
Structure and Operation

Connecting the decoder board to the layout track through the DCC-IN terminal block provides power to the rest of the circuitry via the rectifier bridge and the +5V voltage regulator. The DCC signal, carrying command packets from your command station, is also taken to the Arduino Nano 3 module via a 6N137 optoisolator chip – a device which protects the microprocessor and associated circuits from the potentially destructive DCC track voltage (approximately 14 Volts AC).



Four three-pin headers provide connection to small analog servos such as the Tower SG90, and a dual-row 10-pin header allows connection of the separate keypad board, when required, to set up the servo operational characteristics. There is also a small two-pin programming header to which you fit a shorting link when you wish to set DCC addresses for each of the attached servos. The shorting link can be ‘parked’ on one of the header pins during normal use of the decoder.

The keypad which plugs on to the dual-row 10-pin header holds four pushbuttons, labelled **Se1** (Select), which is used when setting up servo movements, **Op** (Operate), which selects which servo to control, then **L** (Left), and **R** (Right), which make the selected servo move. There are also five light-emitting diodes (LEDs), one red labelled **Se1** (Select) and four green labelled **1**, **2**, **3**, and **4**, which show you which servo is selected and indicate the operation in progress. Details of keypad use will be covered in Part 2 of this article.



Building the Quad Servo DCC Decoder

If you would like to build one or more Quad Servo DCC Decoders (QSDDs), a pair of printed-circuit boards (PCBs) is available from OSH Park, a small company located in Lake Oswego, Oregon. They supply PCBs in multiples of three boards, three decoder PCBs for US\$15.45 and three keypad PCBs for US\$5.25, including free shipping to any destination worldwide. You can see

what OSH Park has to offer, and have a look at the pair of boards, by following these links for the QuadServo-DCCDecoder PCB (https://oshpark.com/shared_projects/CjngC3qb) and QuadServo-Keypad PCB (https://oshpark.com/shared_projects/7ATX5aqB).

If you then want to order a set of PCBs (in multiples of three) click the button labelled “Order Board” next to each board, enter your e-mail address, name, and a password of your choice to establish an account with OSH Park, then follow their ordering process. You can pay either with a credit card or via PayPal. Your boards will be manufactured and delivered within two or three weeks depending on where you are in the world.

If you prefer to use an alternative PCB supplier then, instead of clicking “Order Board”, just click on “Download” to download a copy of the relevant file in Eagle board (.brd) format which you can then send off to your preferred supplier.

Please note that neither A-Train Systems nor myself have any connection with OSH Park other than as a very satisfied customer of their services.

The parts required to build a complete QSD (decoder plus keypad), including cables, are listed in the tables below –

Part - Decoder	Reference	Quantity	Value
Diode Rectifier Bridge	BR1	1	W005G
Capacitor - Polyester	C1, C3, C5	3	100nF
" - Electrolytic	C2	1	220uF 35V
" - Disc Ceramic	C4	1	270pF
Resistor – Metal Film, 0.25 Watt	R1	1	1K2
" "	R2, R4 - R7	5	10K
" "	R3	1	4K7
Diode	D1	1	1N4148
Voltage Regulator - 5 Volt	VR1	1	LM7805
Optoisolator	OK1	1	6N137
Arduino Module	M1	1	Nano-3
Terminal Block - 3.5mm pitch	DCC-IN	1	1 x 2
Pin Header - 0.1" (2.54mm) pitch	JP1 – JP4	4	1 x 3
"	JP5, JP6	2	1 x 5
"	JP7	1	1 x 2
Heatsink – to fit TO220 package		1	
USB Cable - A Plug to Mini B Plug		1	

Part - Keypad	Reference	Quantity	Value
Resistor – Metal Film, 0.25 Watt	R1 – R5	5	220R
Light-Emitting Diode - Red	LED1	1	3mm Red
Light-Emitting Diode - Green	LED2 – LED5	4	3mm Green
Tactile Switch	S1 – S4	4	6mm
Socket Header - 0.1" (2.54mm) pitch	JP1 – JP2	2	1 x 5
IDC Socket Ribbon Cable Connector		1	10-way (2 x 5)
IDC Plug Ribbon Cable Connector		1	Box Header (2 x 5)
Ribbon Cable – 0.05" (1.27mm) pitch		As reqd	10-way

Suggested suppliers for the parts listed above are RS Components (<https://uk.rs-online.com/web/>) or Farnell (<https://uk.farnell.com/>) for users in the UK, or Newark (www.newark.com/) for users in the USA (part of the same company as Farnell). Mouser (www.mouser.com/) or Digikey (www.digikey.com/) are alternative sources in the USA, although their prices tend to be a little higher than Newark. Both Mouser and Digikey also have European-based operations, but still tend to have higher prices than RS Components or Farnell.

The Arduino Nano 3 module can best be obtained from one of the many suppliers operating on eBay, although you will have to make your own judgement as regards who will give you the best service, based on their feedback from previous customers.

The table below gives suggested part numbers for each QSDD component from each suggested supplier. Click on the part number to view the relevant webpage with details of the part –

Ref-Decoder	RS Cmps	Farnell	Newark	Mouser	Digikey
BR1	7082668	1861434	70AC6544	625-B380C1000G-E4	B250C1000G-E4/51GI-ND
C1, C3, C5	3121469	2429342	18AC7634	80-R82DC3100AA50J	399-19335-ND
C2	7111264	8126690	62W6211	80-ESK227M035AG3AA	P5166-ND
C4	7167226	2860060	57AC2084	594-S271K43SL0N6TK5R	BC2679CT-ND
R1	1650230	9341226	95W7689	71-CCF071K20GKE36	S1.2KCACT-ND
R2, R4 - R7	1651031	9341110	95W7695	71-CCF0710K0JKE36	S10KCACT-ND
R3	1650319	9341951	95W7764	71-CCF074K70GKE36	S4.7KCACT-ND
D1	7390290	2675146	05AC0533	512-1N4148	1N4148FSCT-ND
VR1	7147780	1467758	89K1366	511-L7805CV	497-1443-5-ND
OK1	8051267	2453244	31Y6274	859-6N137M	160-1791-ND
M1	eBay	eBay	eBay	eBay	eBay
DCC-IN	8971332	3882615	68C9065	651-1985807	277-6043-ND
JP1 – JP4	2518632	1593422	08N6751	517-929834-01-24-RK	3M9457-24-ND
JP5, JP6	"	"	"	"	"
JP7	"	"	"	"	"
Heatsink	7124257	1611415	81F046	532-507302B00	HS115-ND
USB Cable	eBay	eBay	eBay	eBay	eBay

Ref-Keypad	RS Cmps	Farnell	Newark	Mouser	Digikey
R1 – R5	1650814	9341528	95W7736	71-CCF07220RJKE36	S220CACT-ND
LED1	1780909	1581111	14N9386	859-LTL-4211N	160-1139-ND
LED2 – LED5	1808502	1581114	14N9374	859-LTL-4231N-1	160-1958-ND
S1 – S4	3786476	1555982	95M4260	688-SKHAM	450-1650-ND
JP1 – JP2	8277728	1593490	08N6807	710-61301021821	S7108-ND
IDC Socket	8323483	2215247	45W6459	517-D89110-0131HK	732-2102-ND
IDC Plug	6741205	4139045	94F7977	710-61201025821	732-5452-ND
Ribbon Cable	eBay	eBay	eBay	eBay	eBay

Notes :

1. You may be able to source equivalent parts locally at a lower cost, using the details available for each suggested part by clicking on the links above (assuming that you have

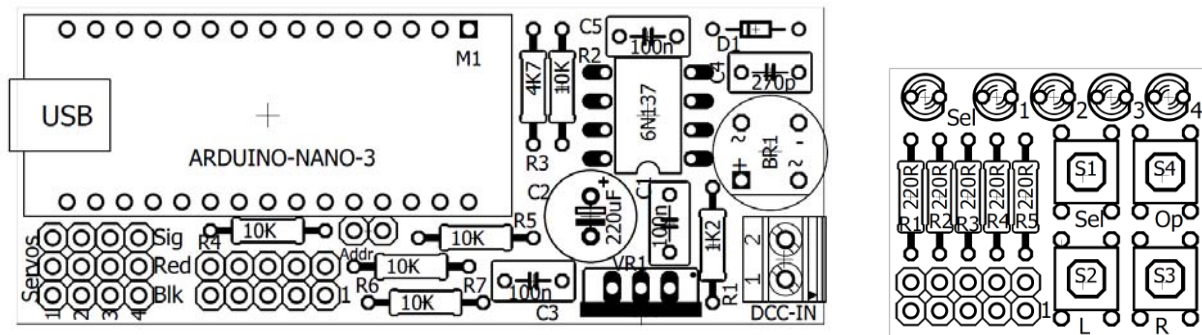
sufficient electronics knowledge to understand the specifications). Although eBay is a very good source for components, especially for small quantities of passive components, headers, connectors or hardware, you need to be wary of very low cost parts, since these are often of low quality or may be manufacturers' substandard rejects.

2. The total cost of parts for a single QSDD should be less than US\$20 (UK£16), including the PCBs, but could be much more if all components are ordered from the major suppliers.
3. Buying electronics components singly or in small quantities is much more expensive than buying in bulk (in quantities of 10 or more), so it is well worth considering carefully at the outset how many modules you might build, and then procuring all of the required components in a single purchase. This will also reduce any shipping charges.

Once you have acquired your PCBs and a full kit of components, the next step is to start the assembly. If you do not have any experience of soldering electronic components then you should first have a look at one or two of the guides available on the Internet (such as at www.makerspaces.com/how-to-solder/) and some of the multitude of videos available on YouTube, although there is nothing to beat getting some copper stripboard from one of the component suppliers and practising soldering wires (and a few spare components) to it before tackling the real module PCBs.

Use resin-cored solder in wire form only – never use solder with an acid flux (as sold for plumbing purposes) – and use a fine-tip soldering iron with a maximum power rating of 25 Watts. All joints should be made as quickly as possible to avoid damaging the PCBs and components. The greatest enemy of electronics is heat.

The position and value of each component is shown on the diagrams below for both decoder and keypad (taken from the PCB design files) –

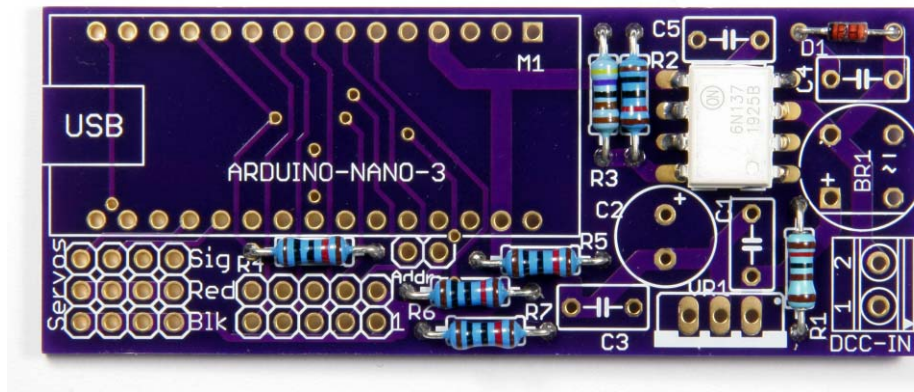


Fit those components with least height to the PCBs first, i.e. the diode (D1) and resistors (R1-R7), so that, when you turn the PCB over and lay it down to solder the component wires on the underside of the board, the components do not fall out of the holes. Ensure that the diode is fitted to the decoder the right way round, as shown on the PCB markings. It does not matter which way round the resistors are placed on the PCB.

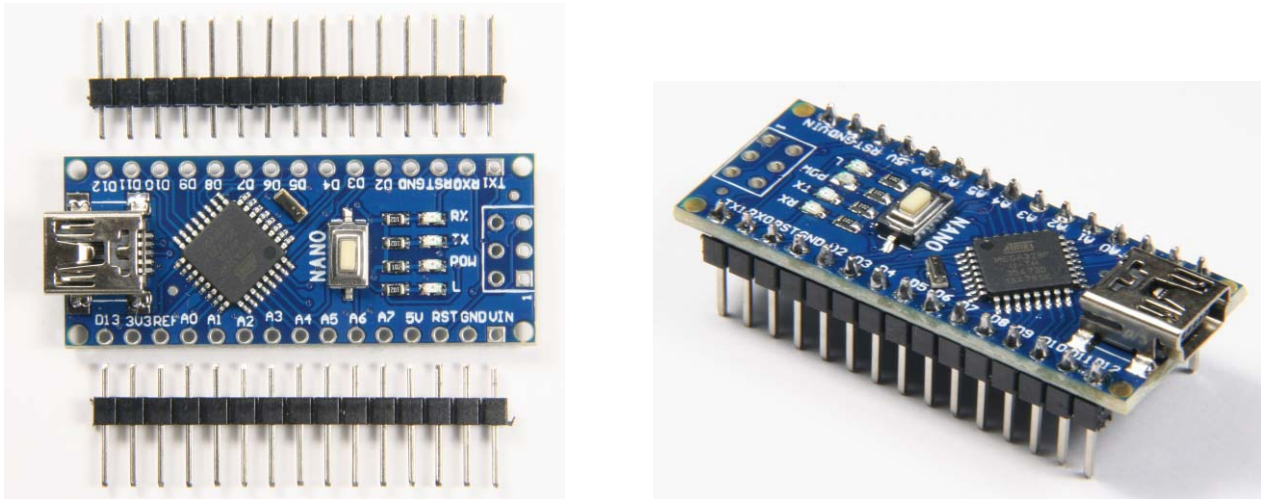
A tip here is to solder just one wire from each component, then turn the PCB over and check that all components are still flush with the PCB. If not, make them so (by briefly melting the one soldered joint while holding the component flat – but watch your fingers – soldered wires are hot!) before soldering the remaining wire(s) of the component. Clip the excess component wires flush with the soldered joints on the underside of the PCB using a small pair of sidecutters.

When you fit the 6N137 optoisolator (as the next tallest components), ensure that the notch or dot at one end of the package is towards the centre of the PCB, again as indicated by the board markings, and that all pins are through the PCB holes, with none having been bent under the device. Solder two diagonally-opposite pins on the chip first, and then check that the device is still

flat on the PCB. If not, it is easy to melt the solder on one corner pin while pushing the device down into the correct position.

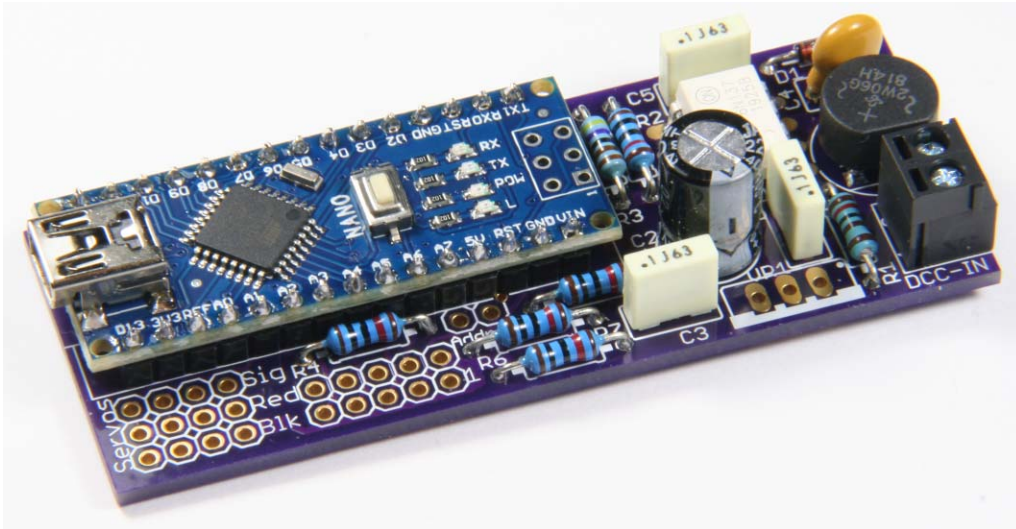


If you purchase the Arduino Nano 3 module 'unsoldered' then you will need to attach the supplied pin header strips to the module before fitting it to the PCB. When doing this, solder the corner pins first, and then check that the headers are fitted square and flush with the module before soldering the rest of the pins, and finally soldering the complete assembly to the PCB. As always, make the solder joints in as short a time as you can, to minimise the heat applied to the module. Note that there is no need in this application to fit the additional 6-pin header, which is usually supplied, to the end of the module furthest from the USB connector.



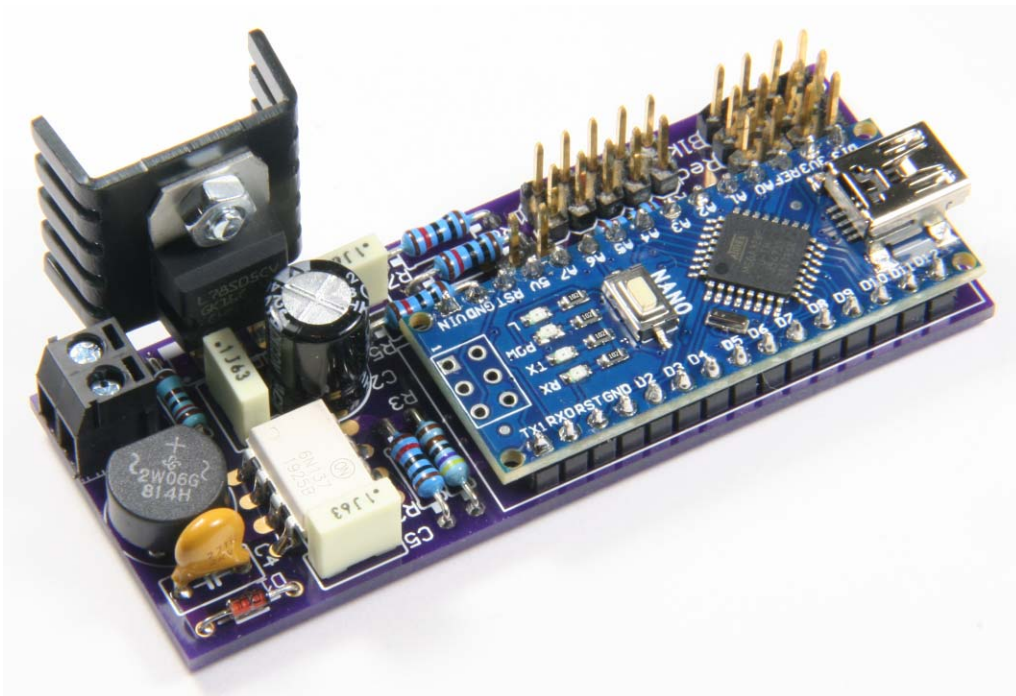
At this stage, before you solder the Arduino Nano to the decoder PCB, you can connect it to one of your computer's USB ports, using the cable listed in the decoder parts table, and check that the module powers up correctly – you will normally see two LEDs lit – one constant (labelled POW) and one flashing (labelled L), depending on the current internal state of the Nano module. You do not need to have any of the Arduino software installed on your computer to perform this check.

Fit and solder the Nano module on to the decoder board, making very sure that the module's USB connector is towards the edge of the board, followed by the capacitors (C1-C5), rectifier bridge (BR1), and terminal block. As the tallest component, the 220uF capacitor (C2) will be fitted last, and you should ensure that it is mounted the right way round, with the stripe indicating the negative (-) terminal towards the nearest edge of the PCB. The other capacitors can be fitted either way round on the board. The flat edge of the bridge rectifier should be next to the terminal block.

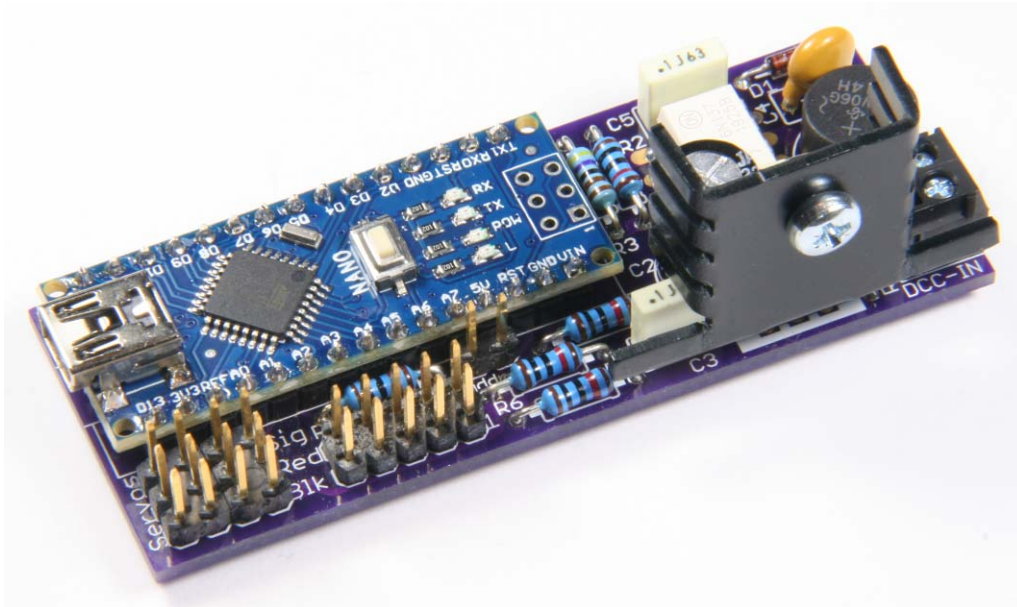


Now fit the pin headers. Although you can buy individual headers to match all of the sizes used, it is much cheaper to buy a single long strip of at least 24 pins, and carefully snap it into the required sizes. Beware of buying the very cheapest headers on eBay, for example, since the plastic used tends to shatter easily, exposing the end pin and often failing to hold it in position.

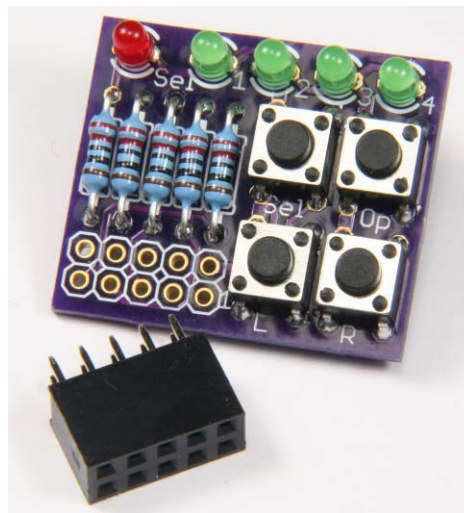
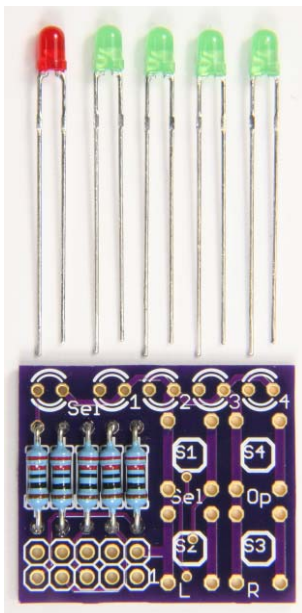
Finally, fit the voltage regulator with the metal tab facing outwards. It is recommended that a heatsink, as listed in the decoder parts table above, is attached to the voltage regulator to reduce its operating temperature to a comfortable level.



The heatsink identified in the parts list can be fitted with the fins either facing away from the voltage regulator, as shown above, or, if you carefully straighten the bottom pair of fins (beware of the soft aluminium cracking and breaking the fins away from the main heatsink body), with the fins facing inwards towards the centre of the PCB.



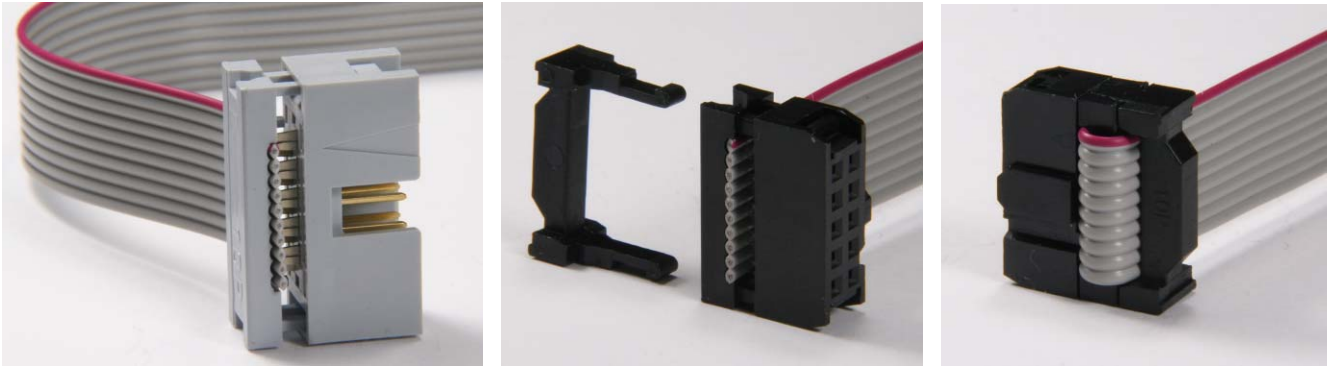
On the keypad PCB, ensure that the LEDs are fitted with the shortest lead towards the printed captions (**Sel**, **1**, **2**, **3**, **4**), and note that LEDs are particularly sensitive to heat, especially when fitted flush to the PCB, so do not linger with your soldering iron. The only point to note is that the 2 x 5 socket header is fitted last, after the four pushbuttons, and fits on the **underside** of the board.



Carefully inspect the completed boards to check that all of your soldered joints are bright and shiny, and that the solder has wicked through the PCB holes to the component side of the board. Check also that there are no solder bridges between copper pads or component pins anywhere on either side of the PCB. Use of a x5 or x10 hand lens or jeweller's loupe is highly recommended for this inspection.

The keypad is designed so that it can be connected to the decoder board either directly or via a ribbon cable. While you can buy suitable ready-made cables, it is straightforward to assemble your own using the listed components. After cutting the 10-way ribbon cable to the required length, and ensuring that the ends are square, place the ends into the insulation-displacement

connectors (IDC) as shown in the pictures below. Firm finger pressure will generally be enough to hold the two parts of each connector on to the cable temporarily, with the tangs just biting into the cable insulation.



Place each connector in turn between the jaws of a vice and carefully apply just enough pressure to close the two parts of each connector on to the cable and complete the joint. Finally, fold the ribbon cable back over the top of the socket connector and slide the strain relief clip into place as shown above (the box-header plug connectors do not appear to be readily available with a strain-relief feature).

You can now plug the keypad on to the 2 x 5 pin header on the decoder, either directly or using the ribbon cable described above, and connect the specified USB lead (A Plug to Mini B Plug) from your computer to the Arduino Nano module. If everything is assembled correctly, the Arduino Nano should light two of its on-board LEDs, one steady and one flashing (depending on the current internal state of the Nano), ready to receive its software program (a 'sketch').

Software for the Quad Servo DCC Decoder

As said earlier, Geoff Bunza's article in the *Model Railroad Hobbyist (MRH)* forum, "SMA20 Low Cost 17 Channel DCC Decoders" (<https://mrhmag.com/node/24316>) inspired the development of this decoder and, for anyone who is not familiar with the Arduino hardware and software which form the basis for all these projects, Geoff's fuller article in the December 2016 issue of *MRH* "A modeler's introduction to the Arduino" (<http://mrhpub.com/2016-12-dec/online/html5/index.html?page=132&noflash>) provides a very good introduction.

The files provided in Bonus Extras for the December 2016 *MRH* (<https://mrhmag.com/magazine/mrh-2016-12-dec/bonus-extras>) include a guide to setting up the Arduino Integrated Development Environment (IDE) on your computer, including links to the Arduino website for downloads (www.arduino.cc/en/Main/Software) and a guide to getting started (www.arduino.cc/en/Guide/HomePage). There are also links to tutorials (https://create.arduino.cc/projecthub/Arduino_Genuino/getting-started-with-the-arduino-desktop-ide-623be4) and setting up the necessary hardware, including the Arduino Nano (www.arduino.cc/en/Guide/ArduinoNano) which is used here in the QSDD.

Once you have the Arduino IDE installed and set up on your computer, and you have mastered the basics by working through some of the examples supplied, you need to use the Include Library function from the Sketch menu in the IDE to add an extra library to the system. This is the NmraDcc library (www.arduinolibraries.info/libraries/nmra-dcc) developed by Alex Shepherd, Wolfgang Kuffer, Geoff Bunza, Martin Pischky, Franz-Peter Müller, and Sven (littleyoda) which lets an Arduino system interpret DCC commands. This library can also be downloaded directly from GitHub (<https://github.com/mrrwa/NmraDcc>) where you can find a lot of examples of the library being used.

You can now download the sketch for the QSDD from the Arduino download section of my A-Train Systems website (www.a-train-systems.co.uk/arduino/download) to any convenient folder on your computer, and then open it in the Arduino IDE.

As well as the NmraDcc library, the sketch calls up the standard Arduino Servo library (already loaded as part of the IDE) to support servo operations rather than the SoftwareServo library used by Geoff Bunza in his applications, since the latter does not allow the degree of control over individual servos which is required for manual setup.

A common failing, noted in the majority of example sketches and libraries that I have come across, is that they contain very little explanation of how they work, or how to use the facilities they provide. Hence, I have tried to include comments on the purpose of each section of the QSDD sketch, and how the various included functions are used. Also, as will be described in Part 2 of this article, you can use the IDE Serial Monitor to display explanatory messages and confirmations of your actions as you proceed to set up the decoder.

Check that the sketch will compile with the included libraries, by clicking the Verify (✓) button on the Arduino IDE toolbar. Any errors are most likely to be caused by the Arduino IDE failing to find the required libraries where it expects them to be, or that the Board, Processor, and Port settings under the Tools menu are not set correctly.

Assuming that all is well, ensure that you have the USB cable connected from your computer to the Arduino Nano, then click the Upload (▶) button to transfer the compiled sketch to the module on the decoder. You should see the Tx and Rx LEDs on the Arduino Nano light for varying lengths of time as the transfer proceeds.

When the upload completes, all five LEDs on the keypad should light up in sequence, and then go out again in the same order, to confirm that the code has been transferred successfully and has run through its normal initialisation process.

Following this, you should see the red **SEL** LED on the keypad light briefly again. This signifies that the default settings for the decoder's configuration variables (CVs) have been loaded into the permanent memory (EEPROM) of the Arduino Nano. This only occurs the first time the decoder is initialised, until such time as you choose to reset all CVs to their default values, as explained in the technical section of Part 2 of this article.

Once the sketch has been uploaded, the QSDD will retain all of the code even when it is switched off, so you can now remove the USB cable from the Arduino Nano before connecting up to four servos to the decoder, ready for adjustment of their throws.

Each servo is plugged on to one of four three-pin headers on the decoder board, making sure that the black or brown wire of each servo is connected to the pin nearest to the edge of the board. Connecting the servo the wrong way round will not do any damage, but the servo will not operate.

Although the decoder alone can be powered from your computer via the USB cable, only modern USB-3 ports have sufficient capacity to power servos as well – an older USB-2 port is likely to be overloaded as soon as you start to drive a servo, an action which will probably reset the decoder and so stop any further drive to the servos.

Hence, you should now connect track power from your DCC system to the decoder's two DCC-IN terminals. If it is not convenient to power up your DCC system at this point, then any power supply of between 9 and 16 volts, either AC or DC, can be used instead – even a 9-volt battery will work.

At this stage, I would recommend that any connected servos are **not** fitted to your layout turnouts or are, at least, disconnected from whatever kind of linkage you are using to drive the turnouts from the servos, since the servo throws (movements left or right) loaded in the software as defaults may not be appropriate to your setup. In extreme cases either the servo or the linkage, or both, could be damaged. It is safer to have the servos free to rotate with, ideally, a

basic actuating arm (“horn”) fitted so that it is easy to see where the servo is currently positioned, as shown in the diagram below, for example.

In this article, servo movements will be described as “left” and “right” as seen when you are looking down on the top of the servo, where the actuating arm is fitted, as shown on the right –

As soon as the decoder has power applied to it, all of the keypad LEDs should light up one after another, and then go out in the same order, and all attached servos should move clockwise to the default rightmost position set in the software (20 degrees right of centre).

To reset the decoder, and so restart the uploaded program (sketch) again, you would normally press the reset button on the Arduino Nano. However, with the keypad plugged directly on to the decoder board, the reset button is not easily accessible, so, as an alternative, just press and hold down the **L** pushbutton on the keypad. Now press the **Op** pushbutton, then release both buttons to reset the Arduino Nano.

To operate a servo, press and release the **Op** pushbutton on the keypad. This will light the green **1** LED to show that servo 1 is selected.

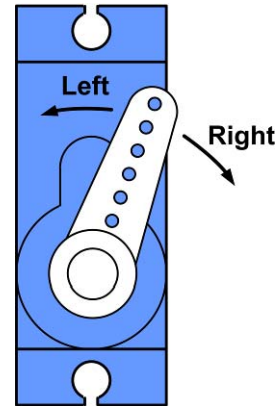
Press the **L** pushbutton to move servo 1 to its left limit position, at the set rate, or the **R** pushbutton to move servo 1 to its right limit position. No movement will result, of course, if the servo is already positioned at the selected limit.

Further presses of the **Op** pushbutton will light green LEDs **2**, **3**, and **4**, in turn, selecting the corresponding servos 2, 3, and 4. When the relevant green LED is illuminated, the currently-selected servo can then be operated by using the **L** and **R** pushbuttons, as described.

A final press of the **Op** pushbutton will switch all LEDs off, stopping manual servo operation.

You can, of course, also operate the servos by sending an accessory command with the appropriate address from your DCC system. With the sketch loaded, the four servos are assigned default addresses of 1, 2, 3, and 4, respectively.

Full details of how to program servo addresses of your own choice into the decoder, and to set the servo throws precisely to suit the turnout linkages on your layout will be given in Part 2 of this article – giving you time in the coming month to build your own Quad Servo DCC Decoder.



Dr Terry Chamberlain

Terry Chamberlain got into model railroading almost by accident in the 1990s when he responded to a request from some modellers in California to build a DCC system based around an Atari personal computer – and he had to build a simple layout to prove that it all worked. Eventually the project evolved into A-Track, a Windows application to provide full computer support for the complete range of NCE DCC systems, with facilities similar to JMRI's Decoder Pro and Panel Pro.



Terry is a professional electronics engineer and spent most of his career in the UK defence industry designing, and managing the development of, large real-time computer systems for the Royal Navy. Now that he has retired he is still hoping to build the logging and mining layout he has been planning for years (after several visits to Colorado) – but keeps getting distracted by new computer and electronics projects for model railroading.