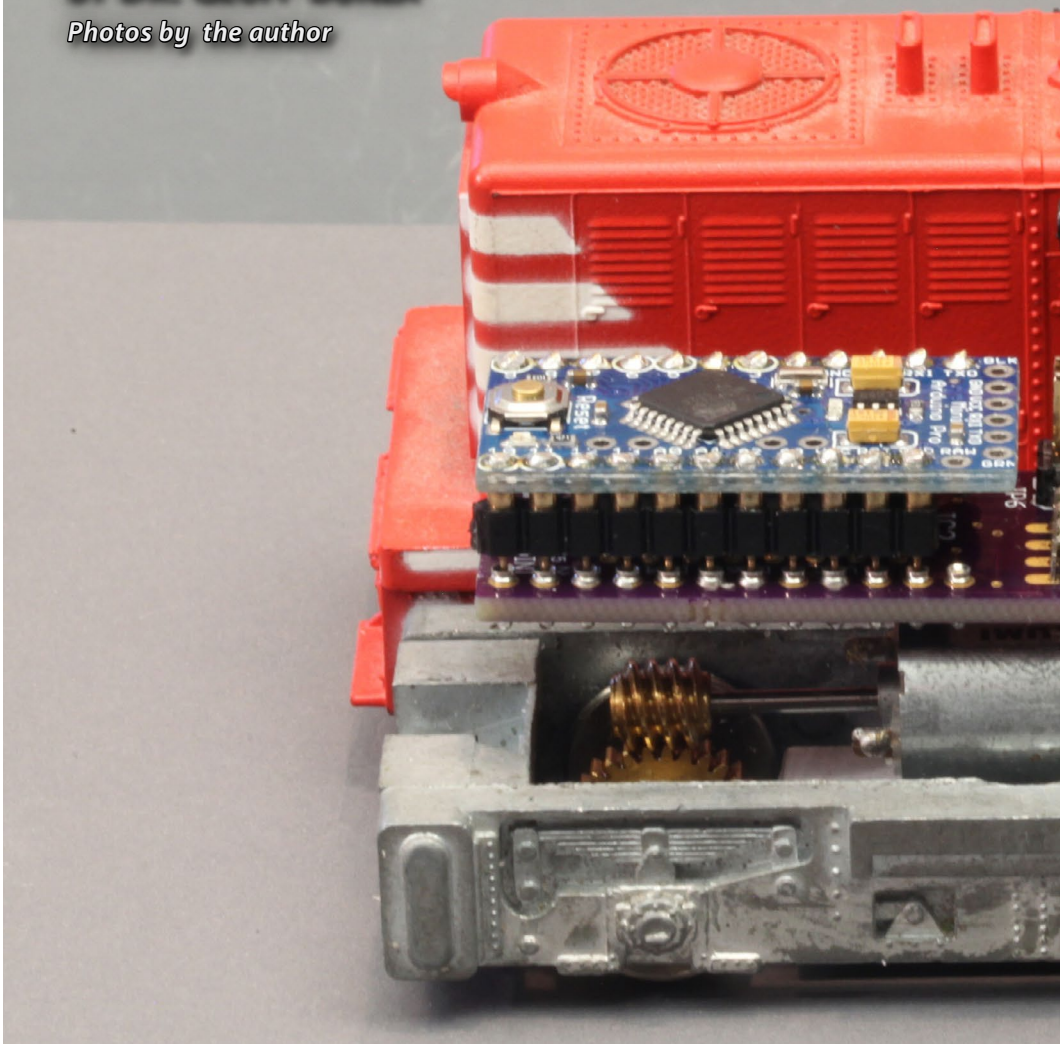
DCC projects using the Arduino

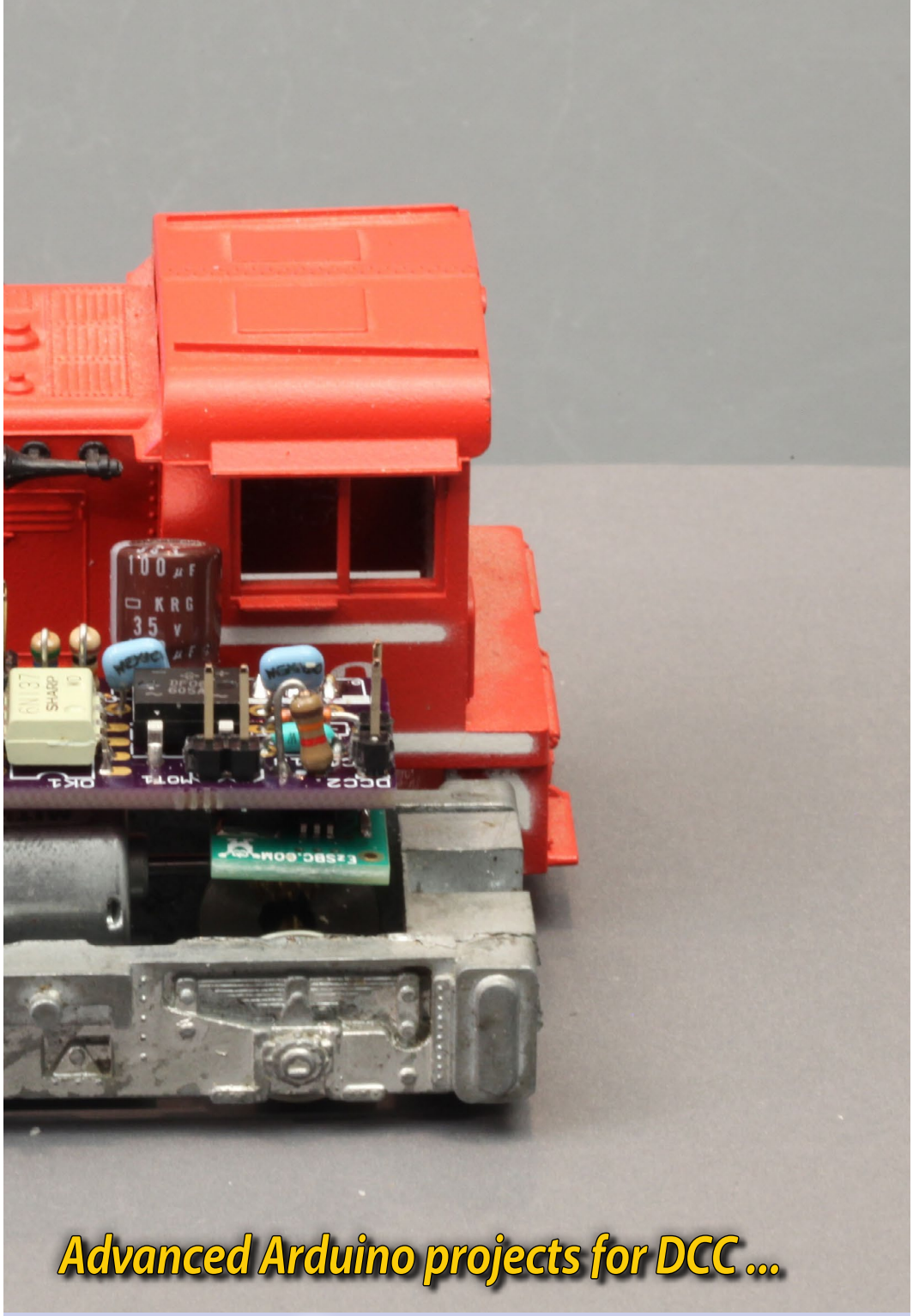
BY DR. GEOFF BUNZA

Photos by the author



1. This is an Arduino-based mobile decoder in a loco. This isn't a project for the faint-hearted, but it illustrates just how incredibly versatile and powerful the Arduino is for model railroaing use.

• INDEX • TABLE OF CONTENTS



Advanced Arduino projects for DCC ...

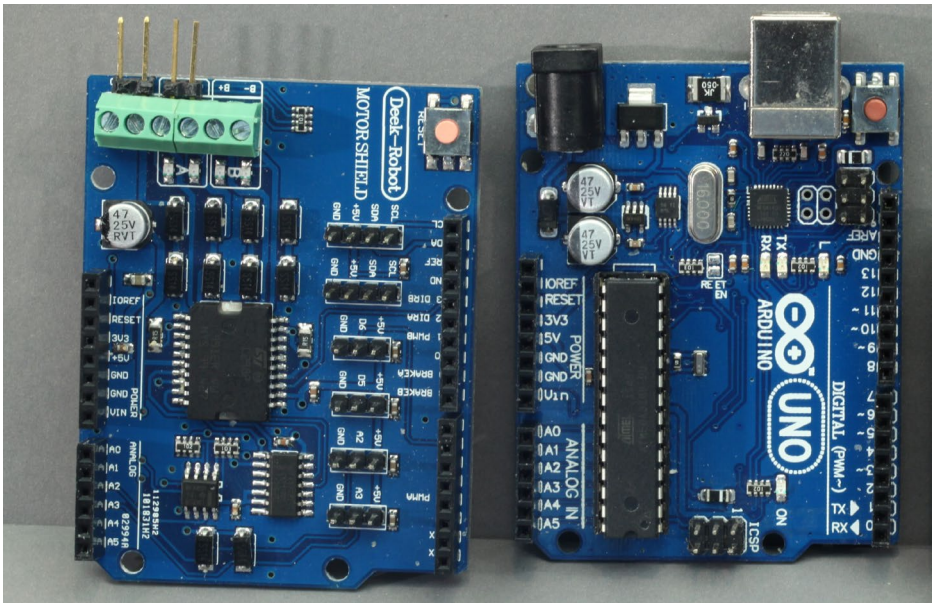
In the December 2016 *Model Railroad Hobbyist*, I wrote an article entitled "A modeler's introduction to the Arduino." This article offered a collection of simple model railroading applications using Arduino controllers. In this article, I'm going into far more advanced DCC applications of the Arduino to show the incredible versatility Arduinos offer.

While this article offers much more complicated projects, it also assumes only a basic knowledge of soldering. I recommend reading my December article first as a good lead-in to the topics here if you'd like to explore more advanced DCC-specific projects with the Arduino. I build on the fundamentals presented in the previous article as I explore these more adventurous DCC projects.

Now, let's see what we can build!

DCC++ base station with a low cost JMRI interface

When I first came across references to the DCC++ project, I was so impressed that I thought more modelers should be aware of its capabilities. Simply put, if you take the two boards pictured opposite [2] (the Uno's motor driver shield and the Arduino Uno), then download and add the efforts of Gregg E. Berman (creator of DCC++, see: sites.google.com/site/dccppsite) and finally add two wire jumpers: ***you get a DCC base station!***



2. Motor driver shield (left) and Arduino Uno (right).

What's in this article?

The first part of this article outlines how a modeler can build a DCC base station with the Arduino, which is just a bit more involved than the lighting projects in the December article.

Following that, we look into a series of very low cost, multifunction DCC decoders that can cope with LEDs, servo motors, DC motors, stepper motors, and play audio. While these are not beginners' projects, they do demonstrate the powerful capabilities of Arduinos.

Do note that the DCC multifunction decoder is a project that modelers from around the world have contributed to from its inception in the summer of 2014. ■

Read the material on each of Gregg's pages and have a look at his YouTube videos. His documentation is quite good.

Are you a fan of JMRI, the free, open source Java Model Railroad Interface (see sidebar)? The Arduino-based DCC++ boards can be controlled with multiple JMRI throttles – yes really!

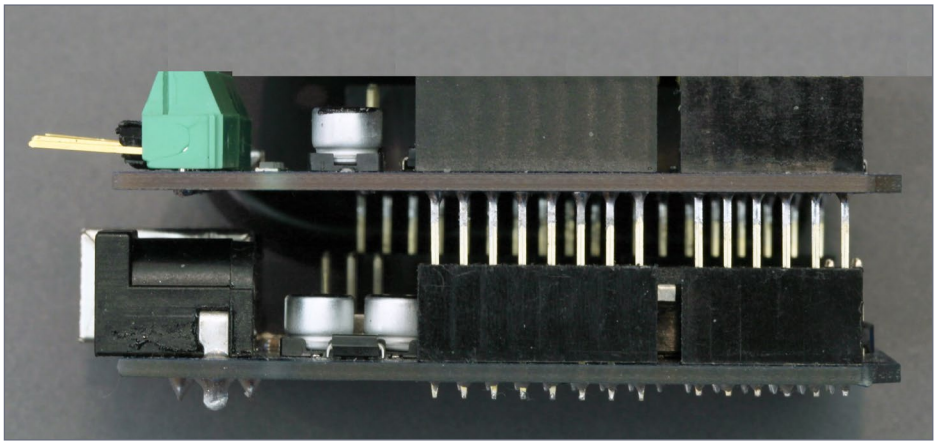
DCC++ first caught my attention when Dave Merrill pointed out its ability to work with JMRI in this MRH forum thread: *DCC++ on Arduino supported by JMRI* (mrhmag.com/node/24757).

Here's how to make a DCC++ base station.

Step 1: Let's build the basic unit. In addition to the Uno control board, you will need an "Arduino Motor Shield R3."

Sources for the UNO include: adafruit.com, sparkfun.com, pololu.com, allelectronics.com, digikey.com, surplusgizmos.com, frys.com, eBay.com, and amazon.com.

Sources for the R3 include: amazon.com (amazon.com/dp/B006U-TE70E), dfrobot.com, and eBay.com (ebay.com/itm/262336202340).



3. Motor driver shield inserted into the Arduino Uno.

The Java Model Railroading Interface (JMRI)

For those who may not know, JMRI is a collection of free software tools that allows connecting a PC, Mac, or Linux computer to many DCC base stations, including the Arduino-based DCC++ system I discuss in this article.

JMRI also includes decoder programming tools, signaling tools, WiFi connected throttles, operations tools, and more such goodies, all free!

I do not cover the details of JMRI DCC installation and operation here. This article does discuss building an Arduino interface to JMRI, however.

Although using the JMRI toolset is beyond the scope of this article, the JMRI website (jmri.org) has a wealth of information including instructions on downloading and setting up the JMRI toolset.

Recent JMRI releases can directly control the Arduino board combo without any other software on your PC. Other modelers, such as the inventive Dave Bodnar, have added even more capabilities.

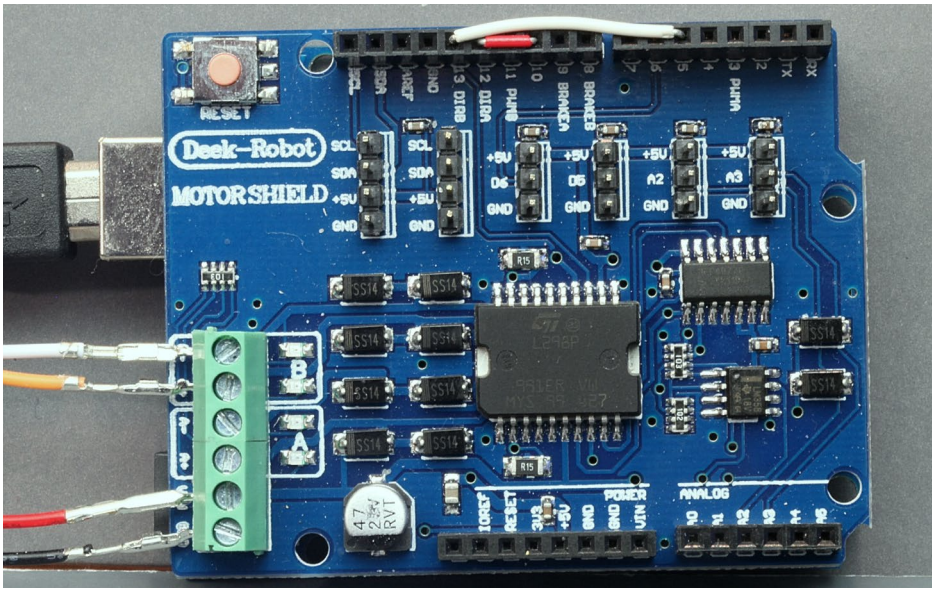
You can follow the DCC++ exploits of Dave Bodnar in this MRH thread: *My Experiments with DCC++* (mrhmag.com/node/25429).

Dave discusses many interesting add-ons, such as adding a booster or doing a remote handheld throttle. Talk about some great stuff! ■

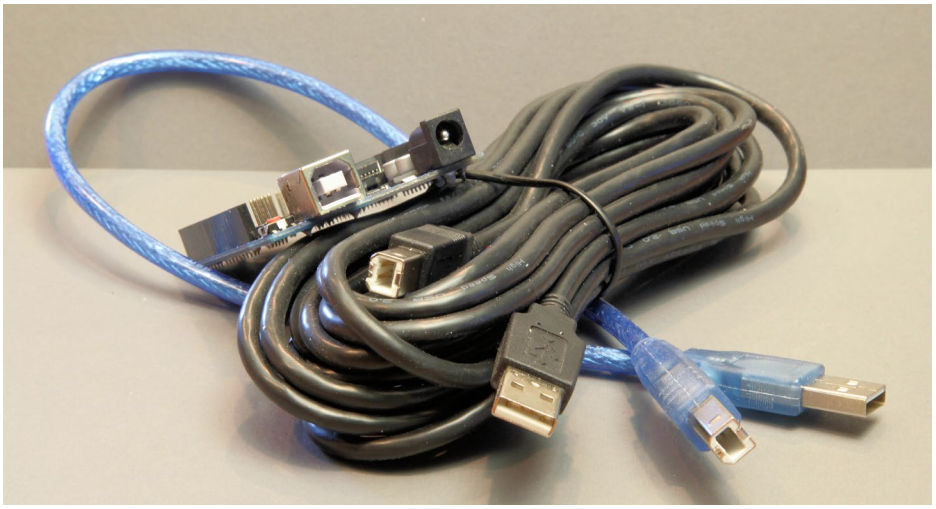
There are other motor shields too, but for the sake of simplicity, only use one like that pictured here [2,3]. This set up allows you to power a two ampere DCC bus.

Step 2: The motor shield board stacks on top of the Uno board, as pictured [3]. Add two jumper wires from D10 to D12, and from D5 to D13, pictured below [4]. Solid number 22 gauge wire does very well, but any wire making a good connection will do. These are the short red and white wires pictured along the top of the board shown below.

Step 3: Plug in the USB AB cable pictured [5] into your Uno and into a USB port on your computer, and you are done with the hardware! The black and red wires connecting to the screw terminals on the motor shield are the ground (minus) and +12 Volt (plus) connections to a DC power supply. The top white and yellow wires are the DCC connections to your track.



4. Red and white jumper wires added to the motor shield (along the top).



5. USB AB cables for the Uno.

Source for suitable power supply: Amazon.com (a.co/iRcqZtn).

This power supply includes screw terminals to ease your power wiring job. This supply will also power the Uno at the high end of its safe input power range.

If you want to run the DCC bus at a higher voltage, follow the advisory here:

github.com/DccPlusPlus/Documentation/blob/master/Motor%20Shield%20Pin%20Mappings.pdf

You will have to cut a jumper wire on the motor shield before applying power.

Step 4: Load the Uno exactly the same way that was covered in my article in the December 2016 issue of MRH. I've also included these instructions in the bonus materials for the March issue.

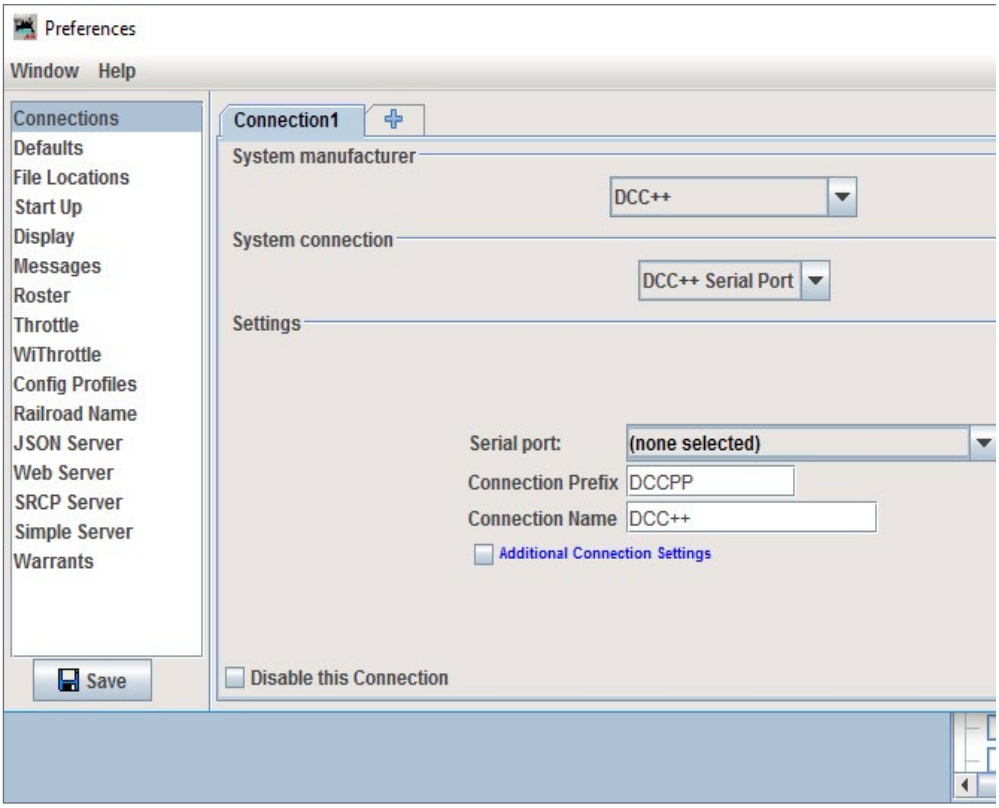
Also from the bonus materials, load the entire **DCCpp_Uno** folder into your **...\Documents\Arduino** folder where your

sketches are stored. Then open the sketch **DCCpp_Uno.ino** and load it into your Uno.

The latest version of the DCC++ base station software can be found here: github.com/DccPlusPlus/BaseStation.

Once you successfully load your Uno you should close down the Arduino editor (IDE).

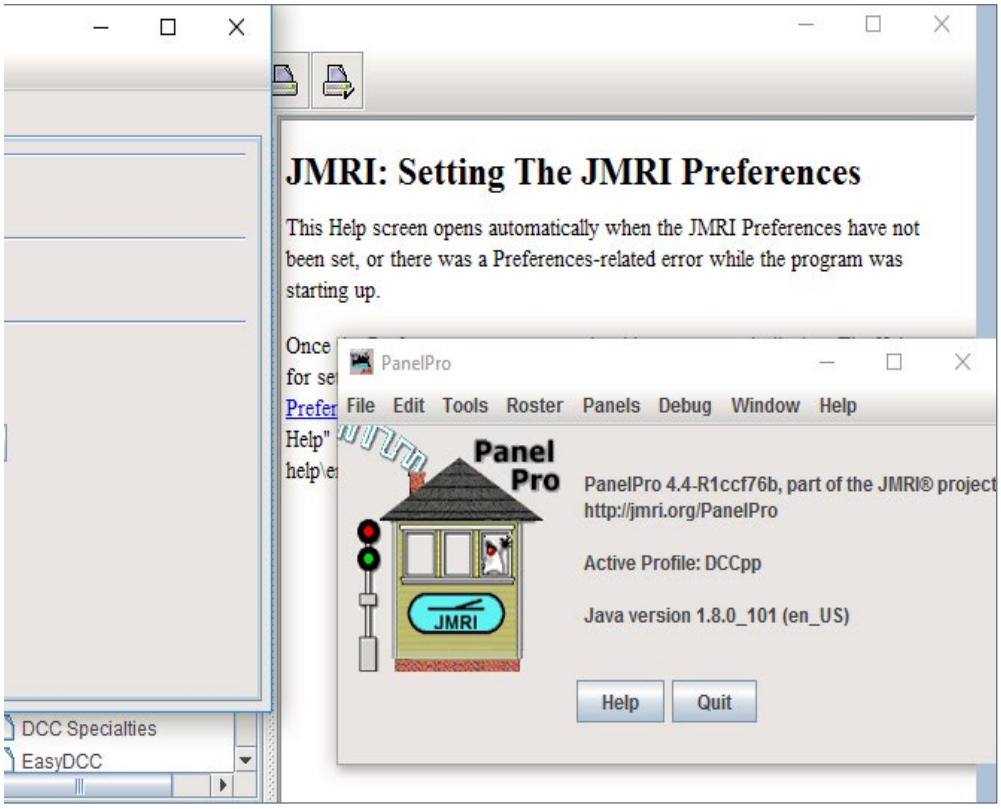
Step 5: Download and install (if not already using it) the latest production release of JMRI from jmri.org/download/index.shtml.



6. JMRI preferences set up screen for DCC++.

You will need to configure the connection to your Arduino/ DCC++ controller by setting up the connection in JMRI preferences: Choose System Manufacturer: DCC++, and choose System Connection: DCC++ Serial Port [6]. This is the same connection you used to download your Uno sketch, and it's the same Serial Port – COMxx from your Uno connection.

Once you have completed this set-up, you can open up a JMRI throttle and run some trains!



This is a simple, low cost, and effective interface to JMRI and it supports multiple throttles and decoders. It also enables a modeler to easily set up and operate a secondary DCC bus which might be used to control a trolley, a mining line, lighting and animations operated via DCC decoders, or even a separate signaling system.

Some modelers are even looking to DCC++ as a low cost entry point to a DCC based layout. There is much more to DCC++ and to JMRI, so I encourage you to explore!

Low Cost DCC Mobile/Function & Accessory Decoders

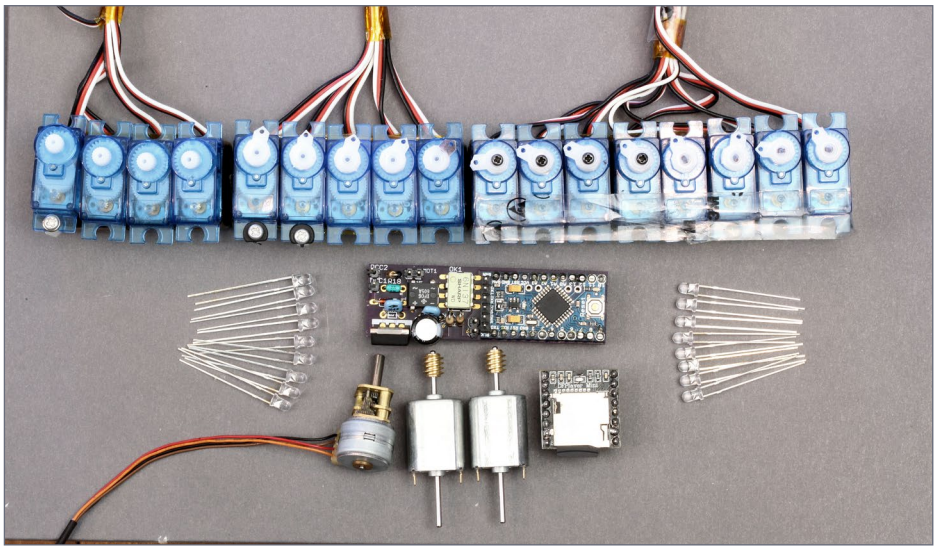
If you are interested in decoders with 17 independent pin functions, where each pin can be set up to perform one of six different operations, all for a cost of $5-$8, then read on!

In this section, I describe a family of DCC decoders that you can build yourself by loading one of 18 pre-configured Arduino sketches. Those who have followed my work before should take note of two new decoders – one for generating audio, and the other for stepper motor control.

Like most decoders, you will be able to customize these further by changing CV values in the decoders. Beyond controlling LEDs and two motors, you can use these to control up to 17 servo motors, a stepper motor, pairs and groups of LEDs, and generate a multitude of sounds.

First, I'll cover the range of decoders that you can build, then I will move to focusing on the basic construction of the decoder hardware. Last, I cover the technical details of the decoder operation.

How it all started: Back in August of 2014 I proposed a nice, simple idea in a blog post to control animations. Commercial DCC decoders and Arduinos had already been used to create animated scenes like this [8, opposite]:



7. An example of devices these Arduino-based decoders can drive.

Playback problems? Click here ...

8. A DCC decoder-based animation.

But with a need for more customized control of LEDs and motors for animation, I looked for more alternatives. With the encouragement, suggestions, and interest from many modelers, the features and functions naturally evolved, or should I say, got completely out of control!

What can you do with a DCC decoder?

After reading my December 2016 MRH article, "A modeler's introduction to the Arduino," and the DCC++ project description in this article, you may be realizing it's the sketch we load into the Arduino that determines the behavior we see at its pins.

The Arduino-based decoders I'm about to describe are based on essentially the same hardware (although we can consider variations here too). With what's available, you can configure a decoder to have a variety of characteristics and either be a mobile/function decoder or a stationary accessory decoder.

See the sidebars "Arduino mobile DCC decoder sketches" and "Arduino accessory DCC decoder sketches" for an overview of the sketches currently available. The description lists the name of the sketch on the upper left which needs to be loaded into your Arduino. All these sketches and libraries listed are included in the subscriber bonus materials for the March issue.

Decoder construction

These DCC decoders were originally designed for animation applications, so cost and size were very important criteria. The availability of very low cost ($2-$6) small Arduino boards (Pro Mini [11]) is an obvious benefit. Alex Shepherd's excellent **NmraDcc** library for the Arduino sealed the decision for me.

Arduino mobile DCC decoder sketches

The following is a list of Arduino mobile decoder sketches that can be used with a DCC system.

SERVO / LED CONTROL

Dec_7Serv_10LED_6Ftn	**7 Servo 10 LED 6 Function**
Dec_10Serv_7LED_6Ftn	**10 Servo 7 LED 6 Function**
Dec_13Serv_4LED_6Ftn	**13 Servo 4 LED 6 Function**
Dec_15Serv_2LED_6Ftn	**15 Servo 2 LED 6 Function**

These are all configured as Mobile/Function Decoders controlling a group of servos and LEDs – the number of each is in the sketch name. Each pin can be set to perform one of six functions: LED on/off, LED blink, Servo motor control, dual LED blink, Pulsed output, and LED fade on. Blink rates can be changed, servo start, stop, and rate of travel can be changed, and pulse duration can be changed all with DCC CV settings per pin. All preset configurations can be reconfigured with ops-mode DCC programming. To be clear, one of these decoders can control 17 servo motors, or 17 LEDs, or any combination thereof.

Dec_17LED_1Ftn **17 LED ON/OFF Control**

This is the simplest decoder (internally). Each DCC Function only turns an LED on and off. This is included for those who might want to learn about the internal operation of the decoder.

Dec_17LED_6Ftn **17 LED 6 Function**

Each DCC Function turns an LED on and off, but each pin can be reconfigured to one of six functions.

Dec_Dir_and_Fade **17 LED with Dual Direction Control & FADE**

Each DCC Function turns an LED on and off based on direction of travel, with fade on and fade off. This is a mobile decoder with a configurable list defining how each of the 17 function pins operate:

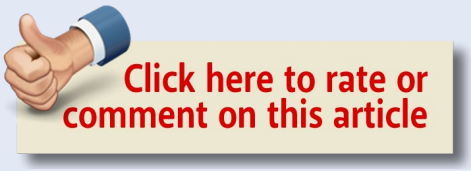
- "0" allows for normal on/off control with fade on and fade off
- "1" allows for normal control when the decoder sees a forward speed setting. Reverse turns the LED off.
- "2" allows for normal control when the decoder sees a reverse speed setting. Forward turns the LED off.

```
byte LED direction [] = {0,1,2,0,1,1,1,1,2,2,2,2,0,0,0,0,0};
//0=On/Off, 1=On Forward, 2=On Reverse
```

Dec_SMA12_LED_Groups **LED group control for euro signaling**

Each DCC Function turns an LED on and off. This is a mobile decoder with five pin sets of arbitrary lighting functions, set in 4-function groups with fade on and fade off. Requested to control German signals with perhaps other uses as well:

- F0-F3 controls preset light group pins D3-D7
- F4-F7 controls preset light group pins D8-D12
- F8-F11 controls preset light group pins D13-D17.



MOTOR CONTROL

Dec_2MotDrive_12LED_1Srv_6Ftn **Dual motor drive, 12 LED 6 Function**

This is a "mobile/function" decoder that supports simple speed control via throttle speed setting for two motors. Motor selection is through motor select Function 13 (Motor1) and Function 14 (Motor2). Motor speed for each can only be changed if the corresponding Function is on (F13 and/or F14).

Motor speed is maintained if the corresponding Motor select function is off. Thus, each motor can be controlled independently and run at different speeds. The other 12 functions are configurable but are preset for LED on/off control.

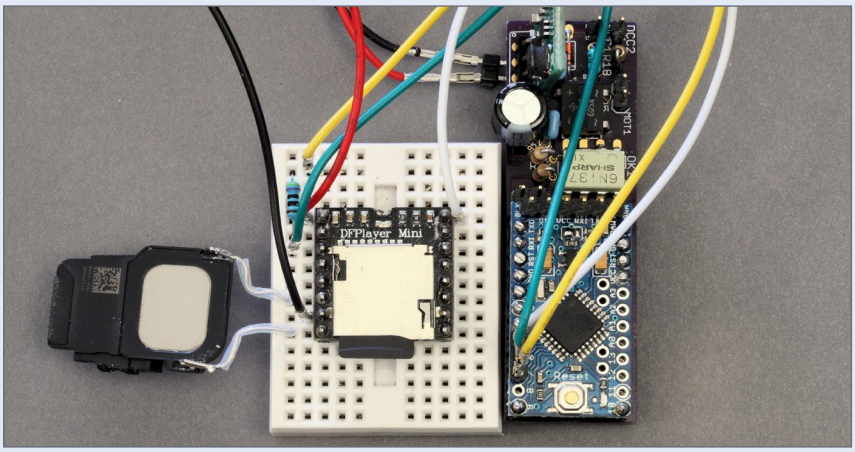
Please note, time dependent functions like servo control and motor speed control interact. Function10 is pre-configured to operate a single servo. I have tested servo operation simultaneous with motor speed control and it worked, but motor timing was affected.

I am using this with small motors (50ma drive) in situations where such timing is not at all critical. Developing better timing control is left as an exercise for the reader.

Dec_2Mot_10LED_Audio_6Ftn **Dual motor drive, Audio Output, 10 LED 8 Function**

This is a "mobile/function" decoder that adds audio play to dual motor control and LED functions. Audio tracks or clips are stored on a micro SD card for playing, in a folder labeled mp3, with tracks named 0001.mp3, 0002.mp3, etc.

F0 is configured as an on/off LED function, F1-F5 play audio tracks 1-5 respectively. F6 plays a random selection in random order of tracks 1-6. F7-F9 control LEDs on Pro Mini Digital Pins 11-13.



9. Audio setup with an Arduino DCC mobile decoder.

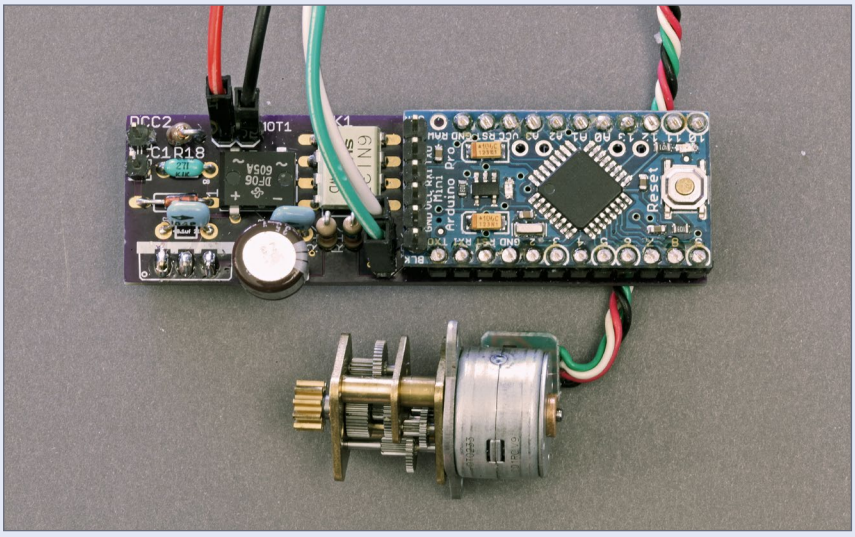
Simple speed control is made via throttle speed setting for two motors. Motor selection is via motor select Function 13 (Motor1) and Function 14 (Motor2). Motor speed for each can only be changed if the corresponding Function is on (F13 and/or F14).

Motor speed is maintained if the corresponding motor select function is off. Thus, each motor can be controlled independently and run at different speeds. The other functions are configurable but are preset for LED on/off control.

DFPlayer Audio Module Wiring to Decoder (Dec_2Mot_10LED_Audio_6Ftn)

DFPlayer

Pin	Other	Pro Mini Pin
1		+5 volts / VCC
2	470 Ohm ¼ Watt Resistor	D7
3		D6
6	8 Ohm Speaker	
7		GND (Ground)
8	8 Ohm Speaker	
16		D5



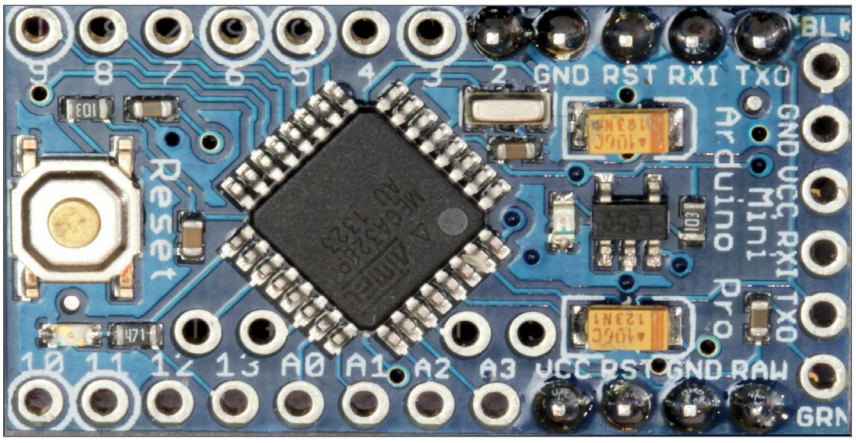
10. Stepper motor connections to an Arduino DCC mobile decoder.

Dec_Stepper_8Ftn **Single Stepper Motor Control**

This is a "mobile/function" decoder that controls a single four wire stepper motor (5 or 12 volt) via throttle speed setting and a multiplier which can be set in CV121. Stepper speed is pre-set in the sketch but can be changed.

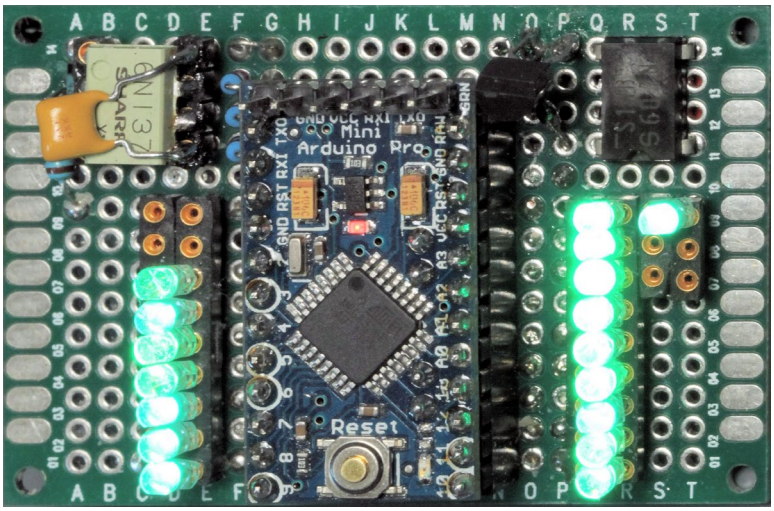
The library also supports setting acceleration/deceleration for the stepper. The other functions are configurable but are preset for LED on/off control. No servo motor control is available. Steppers whose coils need less than 500 ma can be accommodated. Each coil of the stepper attaches to MOT1 and MOT2. You may have to reverse the connections of one or the other until you get the connections right.

The number of steps moved is set by the speed setting multiplied by the contents of CV 121. Every Off to On activation of F2 will move the stepper the specified number of steps, in the direction set by the DCC speed direction. ■



11. Arduino Pro Mini board.

My early decoders were all hand wired on small "perf" boards, since there really were not many components [12]. The entire decoder could be built for less than $5.00! Later, I designed a small printed circuit board to go with the Pro Mini, which added $3.20 to the cost, but made it easier to build – and a lot smaller.



12. Original $5 decoder hand-wired on a "perf" prototyping board.

Arduino accessory DCC decoder sketches

The following is a list of Arduino stationary accessory decoder sketches that can be used with a DCC system.

SERVO / LED CONTROL

AccDec_7Servos_10LED_6Ftn	**7 Servo 10 LED 6 Function**
AccDec_10Servos_7LED_6Ftn	**10 Servo 7 LED 6 Function**
AccDec_13Servos_4LED_6Ftn	**13 Servo 4 LED 6 Function**
AccDec_15Servos_2LED_6Ftn	**15 Servo 2 LED 6 Function**

These are all configured as Accessory Decoders controlling a group of servos and LEDs – the number of each is in the sketch name. Each pin can be set to perform one of six operations: LED on/off, LED blink, Servo motor control, dual LED blink, pulsed output, and LED fade on. Blink rates can be changed, servo start, stop, and rate of travel can be changed, and pulse duration can be changed all with DCC CV settings per pin.

All preset configurations can be reconfigured with ops-mode DCC programming. Accessory decoders only respond to DCC switch commands. To be clear, one of these decoders can control 17 servo motors, or 17 LEDs, or any combination thereof with DCC switch commands.

AccDec_17LED_1Ftn — **17 LED ON/OFF Control**

This is the simplest Accessory decoder (internally). Each DCC switch command only turns an LED on and off. This is included for those who might want to learn about the internal operation of the decoder.

AccDec_17LED_6Ftn **17 LED 6 Function**

Each DCC Switch command turns an LED on and off, but each pin can be reconfigured to one of six functions.

AccDec_7ServoBackandForth6Ftn **7 Servo 10 LED 6 Function**

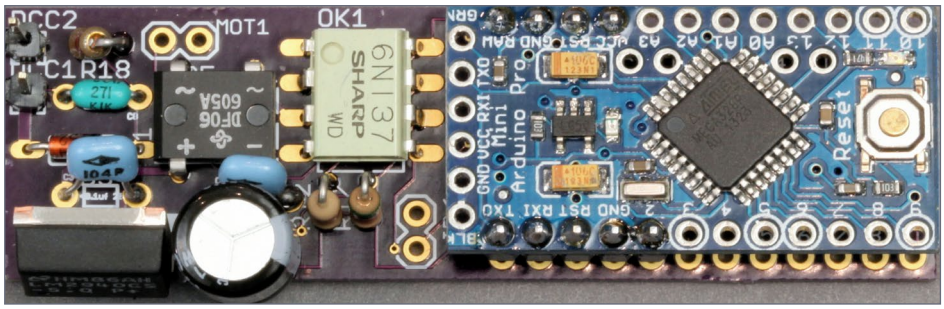
The appropriate servo will travel end to end once, and stop, when the corresponding switch command is set.

The decoder's bill of materials is listed at the end of this article with sources. Order what you need for the decoder versions you want to build, as well as the printed circuit boards (PCBs). There is no special sequence when soldering components. Clear pictures of decoder component placement are provided [13, 14].

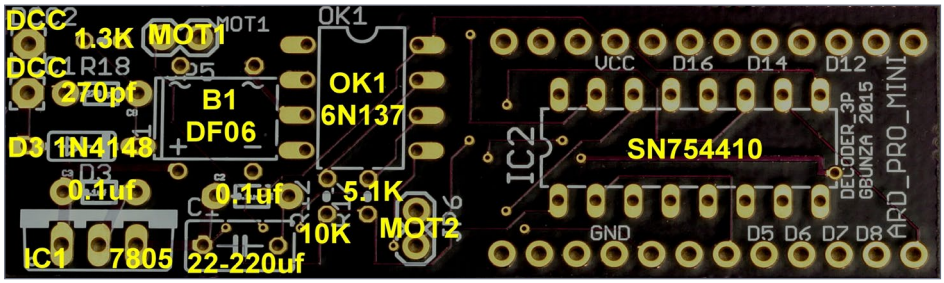
Pay attention to the orientation of the 6N137 and SN754410 integrated circuits, and the DF06 bridge rectifier. Make sure the band/stripe of the 1N4148 diode is placed correctly. Lastly, if you use a polarized capacitor (one that has a "+" and "-") for C1, the positive "+" side is mounted toward the adjacent IC1 voltage regulator.

All the resistors are ¼ watt and are mounted vertically to save space. I add the pins and/or sockets, last. I use a low temperature soldering iron / low wattage soldering iron.

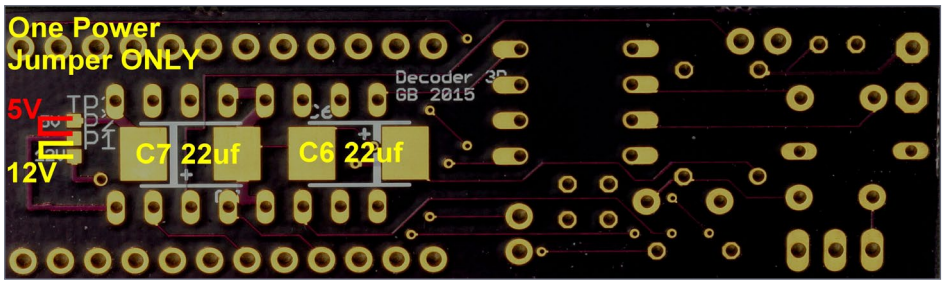
An included printed circuit board makes the construction a bit easier. A dual H-Bridge can be added which allows for decoder control of two motors or a stepper motor. Optionally, the H-Bridge circuit can be omitted.



13. Top mounted decoder components.

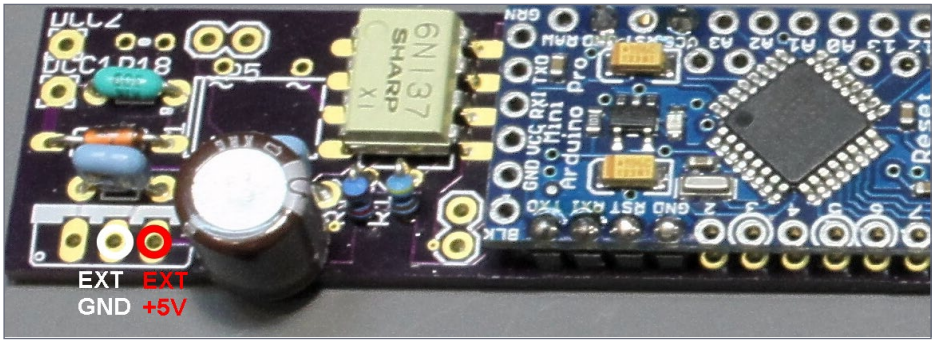


14. Top component placement.



15. Bottom component placement.

The on-board power supply, which powers the board from the DCC bus directly, can also be dropped. This allows for a higher power, local five volt power source from batteries, power adapters, and other power supplies. This board attaches to the Arduino Pro Mini board directly, via header connectors, pins and sockets, or plain wire.



16. Alternate external power connection.

The LEDs and servos are connected directly to the Pro Mini board. The motor connections are located on the add-on board. There is a small set of pads on the underside of the board [15] to select either 5 volt or 12 volt power to the H-Bridge for DC motor and stepper motor power. The middle pad is solder bridged with a small solder blob to the appropriate outer pad.

The Decoder3P is configured to accept a voltage regulator like a L7805 (1.5 amp) regulator. These voltage regulators can get quite warm, so I prefer to use the PSU3-5, a 5 Volt 1 amp cool-running switch-mode voltage regulator, from EzSBC.com. Resistors are 1/4 watt. The value of C1 is listed at 22uf, but I use the largest capacitor I have on hand that can fit in the space with a 25-35 volt rating.

Some of the pictures show a round 100uf 25 volt capacitor that barely fits in the space. The 1N4148 diode can be almost any small signal diode with a 30 volt or more reverse voltage specification.

The SN754410 Dual H-Bridge chip is optional. This is a bi-directional DC driver for two motors or for the stepper motor drive! Before I get the next 500 change requests, the "Dec_2MotDrive_

12LED_1Srv_6Ftn" sketch implements what one might call a mobile function decoder, with the usual throttle/speed control. However, there are no speed tables, jump starts, momentum effects, back emf, doodads, whizzies, or anything else.

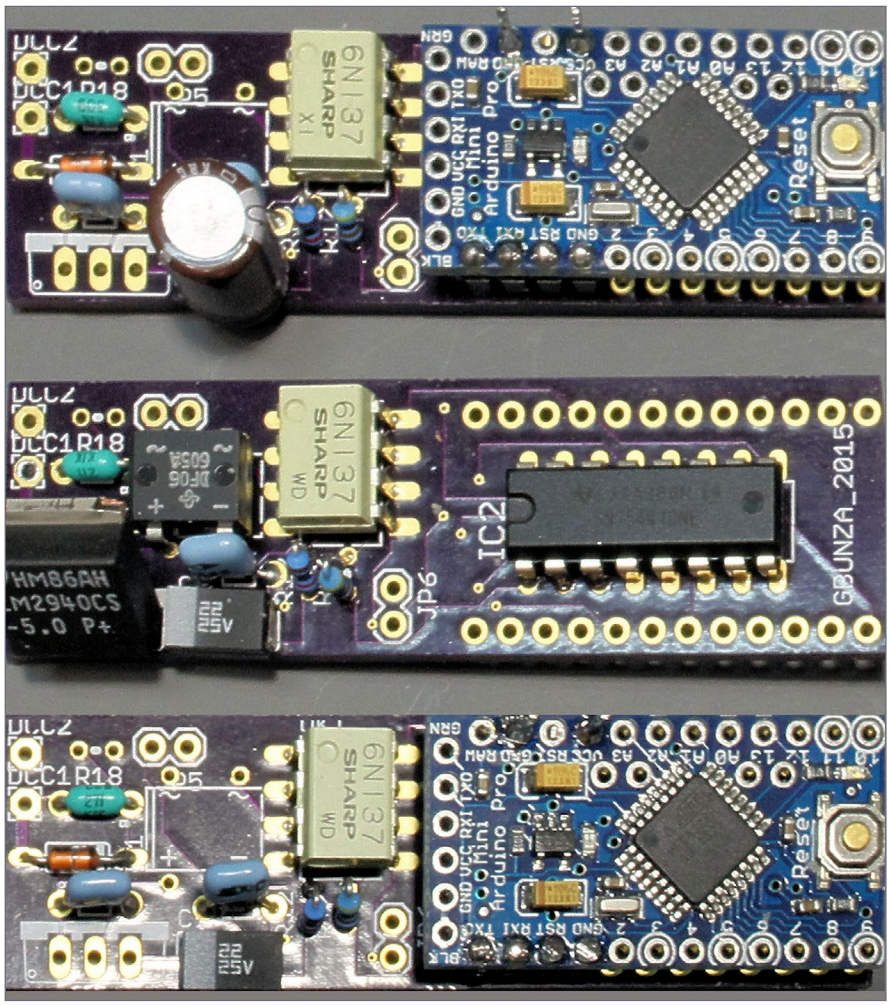
There is only a direct mapping from the 0-127 speed setting to the motor Pulse-Width-Modulation plus direction control of the selected motor. If you do not intend to power any motor(s) you can omit the SN754410, as well as C6 and C7. (C6 and C7 are the surface mounted capacitors on the underside of the driver board.) Don't worry, these are rather large surface mounted capacitors and are easy to solder.

If you power the little board(s) with an external 5 volt supply or batteries, omit B1, C3, and IC1. If you control more than a few servos, you should use a separate 5 volt DC supply due to the high peak power demands of servo motors. See the power attachment points in the Alternate External Power Connection picture [16].

Printed Circuit Boards

The Eagle PCB layout board file (.brd) is included in the additional materials with this article. You can send the board file to any of a multitude of businesses for fabrication. In fact, now I am aware of several people providing this as a service to modelers by manufacturing these and selling them. If I remember correctly there is one shop in Germany, one in the UK and recently Model Railroad Control Systems in California, among them.

I have no financial or business interest with any of them, and I make all my designs to-date available free to everyone. There are likely several hundred modelers that are using these, with over 2000 of these built, by my best estimates.



17. Decoder construction variations.

The boards pictured here were ordered from the board fabricator, OSHPark.com. It is a 2.51x0.77 inch (63.75x19.51 mm) two layer board. OSHPark offers a public service fabricating very high quality, low cost PC boards in low quantities.

Set up an account at OSHPark (really easy) and upload the board file you want to fabricate and specify the quantity (always in

multiples of 3—their rules), either of these cost $9.60 for three (3.20 per board). They accept PayPal and ship internationally.

I have no vested interest in OSHPark – I'm just a satisfied customer. Please feel free to use whatever fabricator you know. The bare boards are shipped "panel-ized" so break or cut them apart. Use the labeled decoder pictures [14,15] and/or the component diagram [21] and solder the components to the board. Remember, you do not have to add IC2, the SN754410 motor driver component, if you do not need it.

The Arduino Pro Mini can either be attached by soldering header pins to the driver board as shown in the pictures (the header pins are sometimes included with the Pro Mini), or it may be socketed with peel-away-socket-strips from Allelectronics.com – also shown below [18].

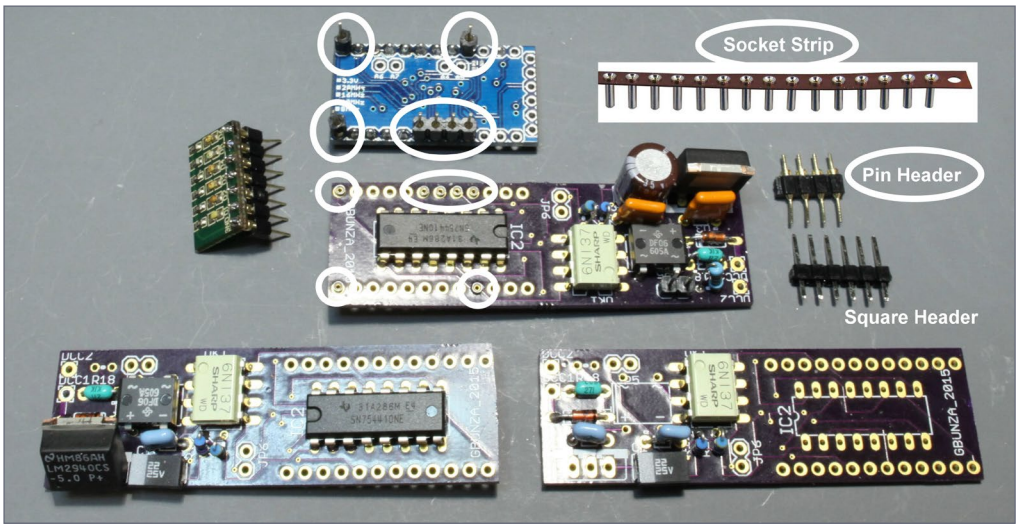
Note: In all cases DCC bus/rail connections are to the DCC1 and DCC2 terminals.

If you intend to use a servo motor attached to Pro Mini Digital Pin 13 – the same pin used to drive the built-in LED – please unsolder the LED, or the LED dropping resistor on board, or cut the trace to the LED to disable it [20]. The LED connection often interferes with the servo control on that pin!

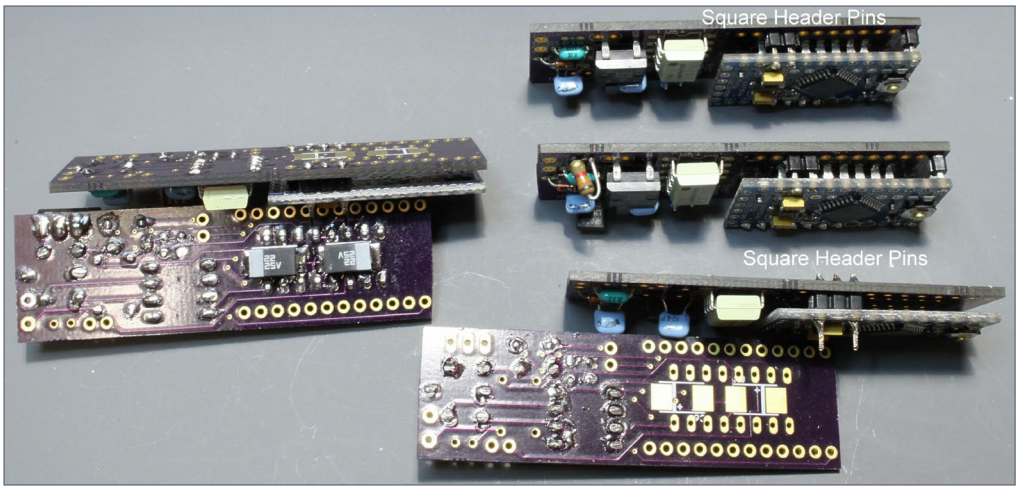
Build the decoder hardware variant of choice, and then look here for information on setting up and loading the decoder sketch (program) of your choosing. With that said, a step by step cookbook for loading the decoder, oriented to the modeler, can be found here: "Starting from Scratch with an Arduino Pro Mini" (mrhpub.com/2014-11-nov/land/#99). It is also provided in the additional materials with this article.

In the decoder examples, servos are preconfigured on the lower numbered pins, contiguously, followed by the "LED" drivers. The

file names say it all, so pay attention. Remember, you can configure each pin to do what function you would like, including a 17-servo driver. Load them into your Pro Mini and you are good to go!



18. Socketed decoders.



19. Pin-soldered decoders.

The libraries and examples are included in the additional materials with this article. Updates can infrequently be found on my MRH blog: mrhmag.com/blog/geoff-bunza or on mrrwa.org in the NmraDcc Library depository.

The **NmraDcc** library, provided in this article's additional materials, in its folder (NmraDcc) should be placed as is in your ... `\Documents\Arduino\libraries\` folder on your computer. Do not modify it.

Final notes

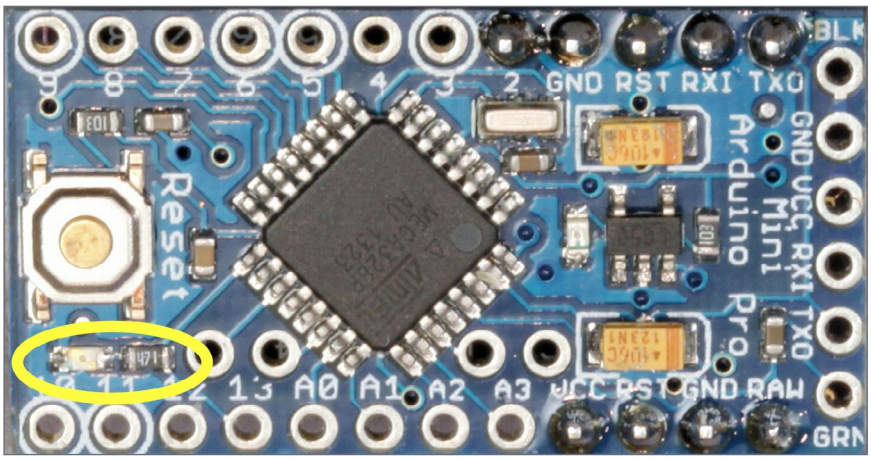
These decoders have been used with a wide variety of commercial DCC base stations and DCC++. A reported common use for these decoders is to run multiple servos to control track switches.

Many of these variations have been specifically requested by modelers from around the world. The two wire DCC bus with power allows for remote control, and coordinated control of lighting and animation. This is why the audio and stepper motor additions were created.

Close inspection of the decoder configurations will show there are hundreds, if not thousands of variations available to the modeler. I rarely use Ops Mode DCC programming to set these up. Rather I configure them in the Arduino editor and simply download the new sketch, and save it for later reference.

Many modelers have taken these sketches as the starting point for their own efforts. I encourage this in every way. It's another way to have more fun with modeling!

There has never been any intention with these projects to create a competitive commercial decoder – and there still is none. I only want a simple reconfigurable decoder that I can customize for my own model work. I hope those modelers reading this can



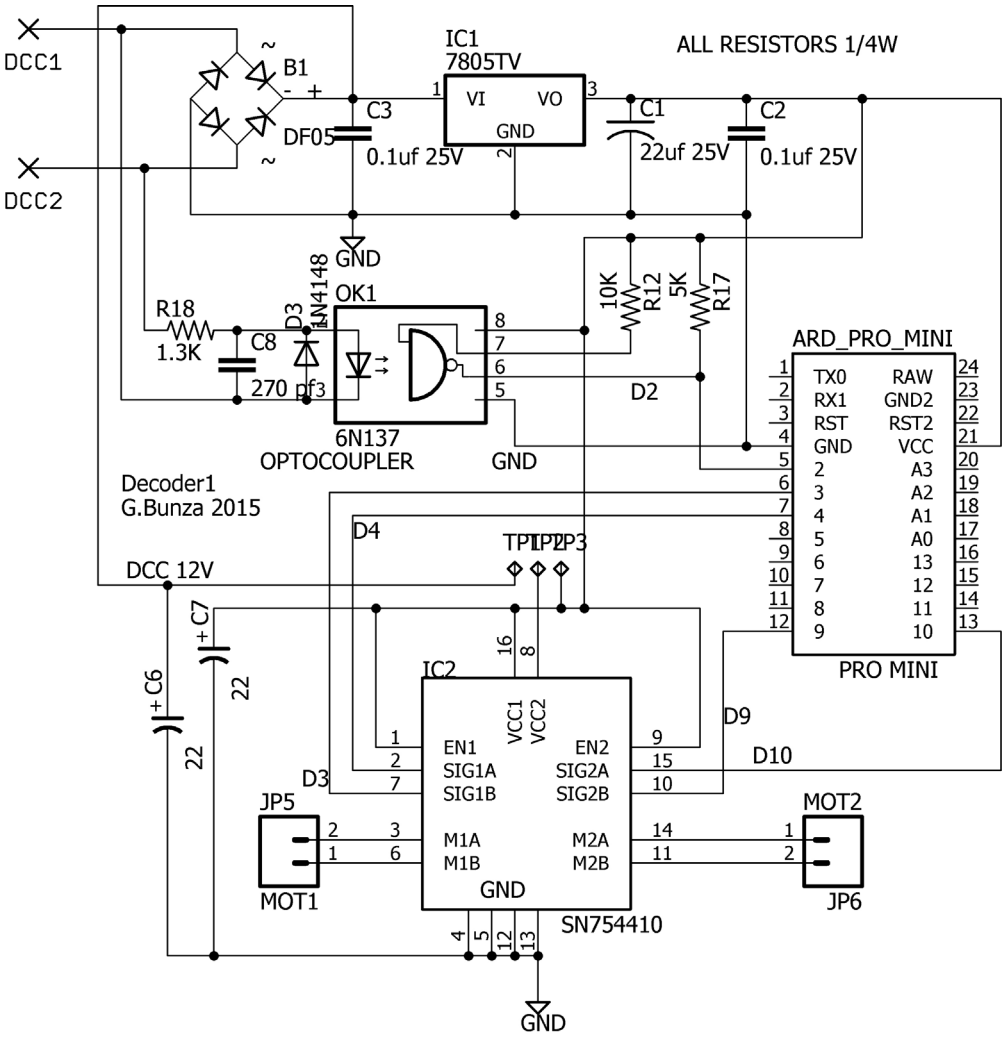
20. Pro Mini LED and/or resistor to be removed.

take advantage of these efforts, and will also share your variations with others.

Many of these have been built, perhaps many more than I am aware of. I doubt very much that I would have pursued all these variations, especially the accessory decoder variants, without the interest and enthusiasm shown by scores of modelers, literally, from around the world. To all of you – many sincere thanks!

Many thanks again to Alex Shepherd of New Zealand for his work on the **NmraDcc** library, and a special note to Franz-Peter Müller for his suggestions on code improvements. ☑

More how-to materials for this article are included in the March issue subscriber bonus downloads.



21. DCC_Decoder3P schematic.

See the next page for a video demonstration of what can be done with the DCC_Decoder3P ...

Playback problems? Click here ...

Geoff Bunza

Geoff Bunza started as a model railroader when he received a Mantua train set for Christmas, at age 6. He fed his interests through college, becoming a member of the Tech Model Railroad Club (TMRC) at MIT while getting his doctorate and three other degrees in electrical engineering. He has collected Lionel HO trains for many years, which spawned his interest in realistic model animation and lighting. Primarily, he models the New York Central Railroad.

Geoff is a member of the New York Central System Historical Society, a life member of the NMRA, and holds an Extra Class amateur radio license. ■

Bill of Materials: Decoder3P, Quantity One

Digikey parts (digikey.com):

OK1	160-1791-ND	6N137 OPTOCOUPLER HS 8-DIP	0.81
B1	DF06M-ND	DIODE BRIDGE 600V 1.5A 4-DIP	0.41
R12	CF14JT10K0CT-ND	RES 10K OHM 1/4W 5% CARBON FILM	0.10
R17	CF14JT5K10CT-ND	RES 5.1K OHM 1/4W 5% CARBON FILM	0.10
R18	CF14JT1K30CT-ND	RES 1.3K OHM 1/4W 5% CARBON FILM	0.10
C2,C3	445-8421-ND	CAP CER 0.1UF 25V 10% RADIAL	0.29
C8	BC1018CT-ND	CAP CER 270PF 50V 5% RADIAL	0.35
C6,C7	478-8312-1-ND	CAP TANT 22UF 25V 10% 2312 SMD	1.22
C1	P13476-ND	CAP ALUM 100UF 20% 25V RADIAL	0.32
	OR		
C1	478-8312-1-ND	CAP TANT 22UF 25V 10% 2312 SMD	1.22
	OR		
C1	493-5914-1-ND	CAP ALUM 220UF 25V 20% RADIAL	0.38
IC2	296-9911-5-ND	SN754410 IC HALF-H DRVR QUAD 16-DIP	2.43
IC1	497-15682-5-ND	L7805 IC REG LDO 5V 1.5A TO220	0.58
	OR		

IC1 PSU3-5 5V 1A Cool-running switch-mode voltage regulator. from EzSBC.com:
www.ezsbc.com/index.php/products/psu3-5.html#.VpGVvV55yMQ

PSIP-80 PEEL-A-WAY(R) MACHINE PIN SOCKET STRIP 2.50/50 socket pins
allelectronics.com/make-a-store/item/psip-80/peel-a-way-r-machine-pin-socket-strip/1.html
These socket strips accept Integrated Circuit DIP packages, or 0.020 wire – like the Tichy phosphor bronze wire.

SHS-40 1 X 40 HEADER, 0.1" SPACING 0.85 per strip of 40
allelectronics.com/item/shs-40/1-x-40-header-0.1-spacing/1.html