

A modeler's introduction to the

Arduino

BY DR. GEOFF BUNZA

Photos by the author



• INDEX

• TABLE OF CONTENTS





• INDEX

• TABLE OF CONTENTS





[Click here for
reader comments](#)

MODEL RAILROADERS THROW SWITCHES ALL the time – to turn on the power, to switch a track, to blow a whistle, to turn a light on or off, and more. But what if you had a little automated help?

Suppose you want to easily set up a route through a yard, or display a sequence of lights on a movie marquee? Maybe you would like to model a welder in action, or to simulate a thunderstorm?

My approach here is to introduce non-technical modelers to some easy but useful projects like these without getting buried in the technical jargon. I'm focusing more on the do-it-yourself basics, and I'm not diving into all the technical details of how it works.

For the purposes of this article, just know it works, let's not be overly concerned with the details of why.

Anyone can put these projects together, some with little or no soldering. You will not need to read a schematic or program a computer. However, you may need some imagination to envision all the possibilities available to enhance your modeling!

Introducing the Arduino

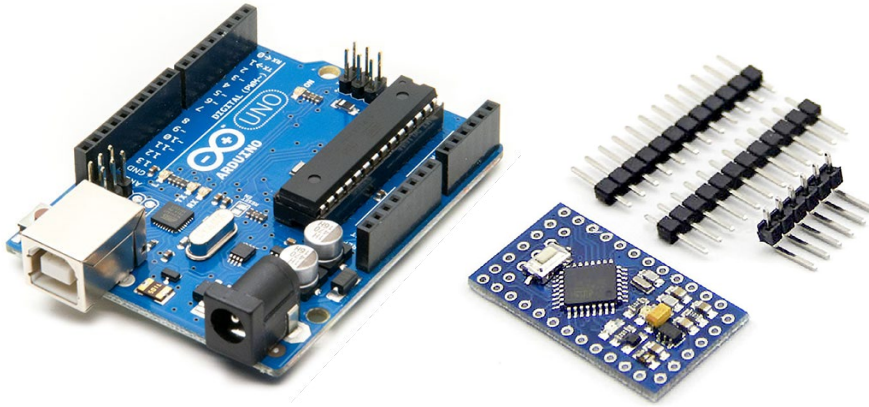
The Arduino is a user-friendly board family for building electronics projects. While physically different, the various boards in the Arduino family all perform many of the same basic functions, controlled by a very small but powerful microcomputer chip.



• [INDEX](#)

• [TABLE OF CONTENTS](#)





Arduino Uno

Arduino Pro Mini

2. Here is the Arduino Uno board and the Arduino Pro Mini board with “header pins” that fit into the board holes. As you can see, connecting to the Uno is easier than with the Pro Mini, but the Mini is far more compact and costs less. Both function the same.

What is the Arduino?

“Arduino” is an Italian name for a group of electronic control boards that are low cost, small, and remarkably useful. While physically different, they all perform many of the same basic functions.

In this article, I use two small boards in the family, the Uno and the Pro Mini [2]. The Uno is easier for a novice to use, but is larger and costs more. At the time of this writing, the Uno costs between \$8 and \$20, depending on where you purchase it.

The smaller Pro Mini will substitute for the Uno without modifying the project instructions and can be found for as low as \$2. The Uno is easier to connect to, so we start with it. ■



• INDEX

• TABLE OF CONTENTS

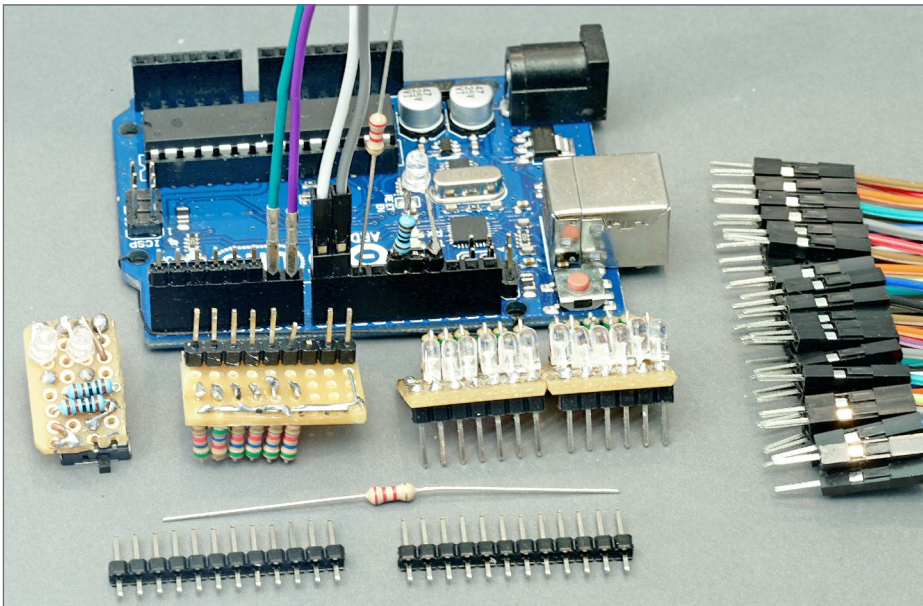


MODELER'S INTRO TO THE ARDUINO | 5

Originally, the Arduinos were created for use as a digital electronics and software teaching tool. All aspects of their design and construction are totally open and available for public use rather than being proprietary. In this article, I use two small processor boards in the family, the Uno and the Pro Mini [2].

An Arduino project consists of both the programmable board and some instructions called “sketches” in the Arduino world. To provide the sketch instructions to the board, you use an app on your home computer called an IDE (Integrated Development Environment) to edit and load the instructions into the board’s memory. See the [bonus downloads](#) for IDE instructions.

The instructions tell the Arduino what “switchable connection points” to turn on or off, and in what order and at what time.



3. The larger Uno board has “header pin sockets” that easily allow plugging in wires and components like resistors to the board. Here I built some other pluggable boards too using LEDs and resistors.



• INDEX

• [TABLE OF CONTENTS](#)



What this article is and is not

Normally, an article on a technical subject like this would start with basic concepts and terminology, and then move on to describing the tools, the process and usage. And of course, there would be plenty of very technical footnotes all along the way.

But this article is for modelers *and we want to have some fun!*

I'm focusing here on interesting projects that take a minimum of effort. Project categories include lighting, servo control, and sound generation. Any project can be just a one-time effort, or together these projects may start you on a new sub-hobby.

For the more technically savvy, Arduino development includes a fully Integrated Development Environment (IDE) that incorporates a GNU GCC optimizing compiler. The IDE is automatically configured for a family of dissimilar microprocessor modules via background scripts and file descriptors. This is all enabled by freely available libraries covering a wide range of drivers and sensors. This platform directly supports C, C++, and object oriented programming.

Are you lost yet?

Rather, this article takes the simpler route of whetting your appetite by showing how to easily use a \$2 device (the Arduino Pro Mini) to align various routes through a yard, ring bells, sequence lights, model a welder, create a thunderstorm, and maybe perhaps bring things to life on your layout that could even impress non-modelers!

Indeed, a tech savvy modeler should appreciate this too. ■



• INDEX

• TABLE OF CONTENTS

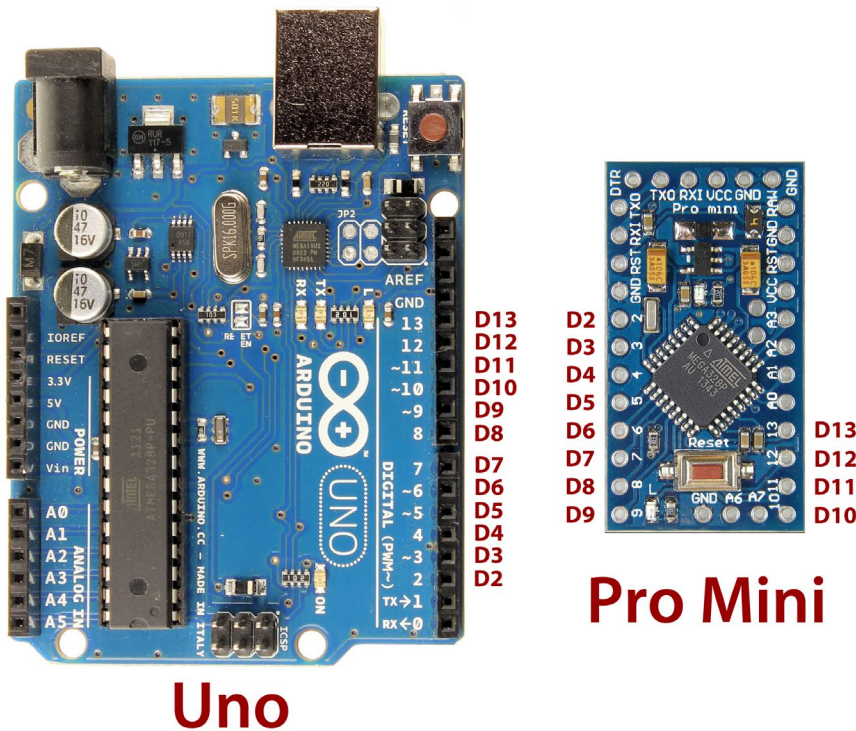


MODELER'S INTRO TO THE ARDUINO | 7

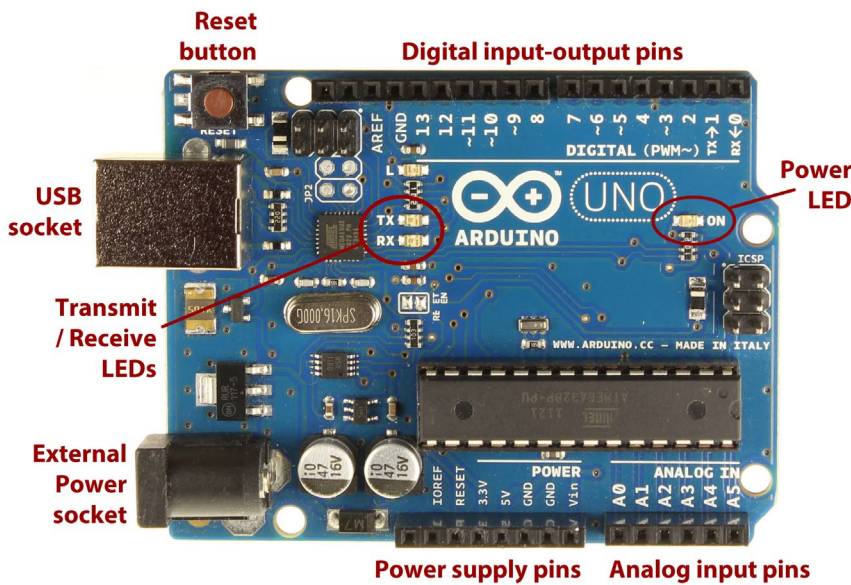
Both the Uno and the Pro Mini can be loaded with the same sketch instructions. Both boards have the same number of switchable connection points (pins) that are consistently labeled the same [4].

It often is easier to test out a project on the larger Uno, and then load the same instructions into the smaller and cheaper Pro Mini for final use.

Let's start by getting familiar with the Uno. The Uno is the larger of the two boards and was designed for ease of use and simple interconnection with other companion boards called "shields."



4. The Uno and the Pro Mini have the same digital pin numbers, as you can see here. This allows starting a project using the easier-to-test-with Uno and then copying the final tested instructions into the Mini for deploying the finished project.



Arduino Uno board "basic layout" Version R3

5. The Uno has this basic layout: a USB socket, external power socket, reset button, some LEDs on the board, and a number of connection points using header pins (digital, analog, and power).

The Arduino Uno

Notice the Arduino Uno has connection points around the edge as female pin sockets [3, 5] into which you can plug wires, components like LEDs and resistors, and special "shield" board pins to form a stack [6, 7].

Shield boards neatly enhance the capabilities of the Uno. Two common shield boards are a motor control shield and a sensor shield.

The motor control shield allows the Uno to power high current devices: I will provide an example of its use later in a sample project.

The sensor shield actually has no sensors itself. Instead, it provides a connection to all the switchable pins on the Arduino Uno



• INDEX

• TABLE OF CONTENTS



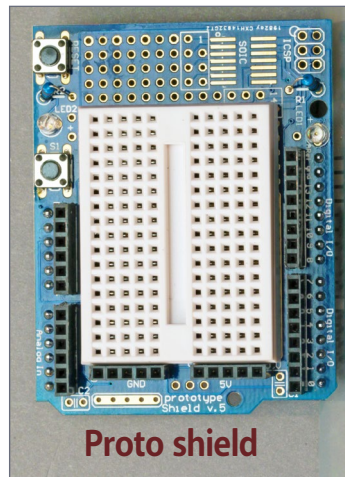
MODELER'S INTRO TO THE ARDUINO | 9

side by side with 5 Volt and Ground (GND) pins, allowing easy connection to a number of special sensor modules. But it also turns out, this is precisely the right arrangement for easily connecting with servo motors too!

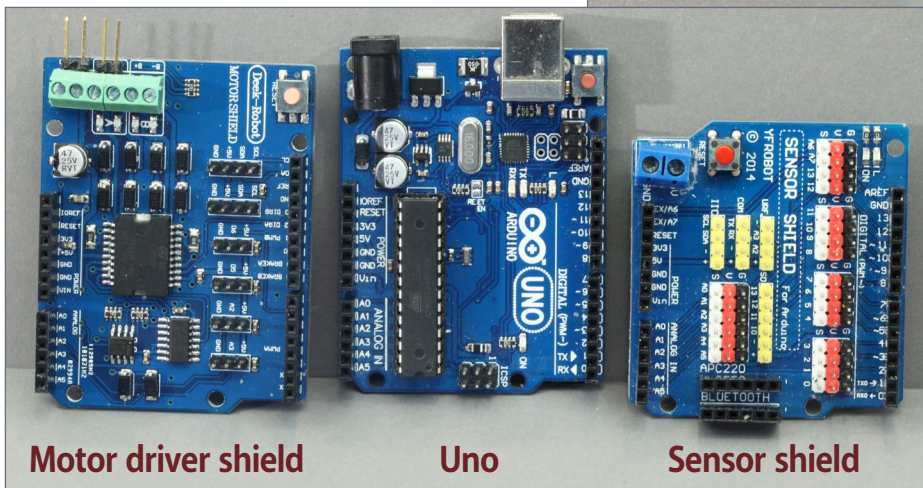
You can insert wires and component leads directly into the Uno sockets or use “header pins” and solder connecting wires to the pins. The header pins allow disconnecting or changing your wiring fairly easily. Using header pins is quite reliable and I heartily recommend their use [3].

The website Pololu.com (pololu.com/category/71/wires-with-pre-crimped-terminals) offers a wide selection of wires with the appropriate pins already attached. Or you can search for “jumper wire” to find other sources.

6. Here are some of the the add-on “shield” boards that you can plug into the header pin sockets on the Uno board. See text for details.



Proto shield



Motor driver shield

Uno

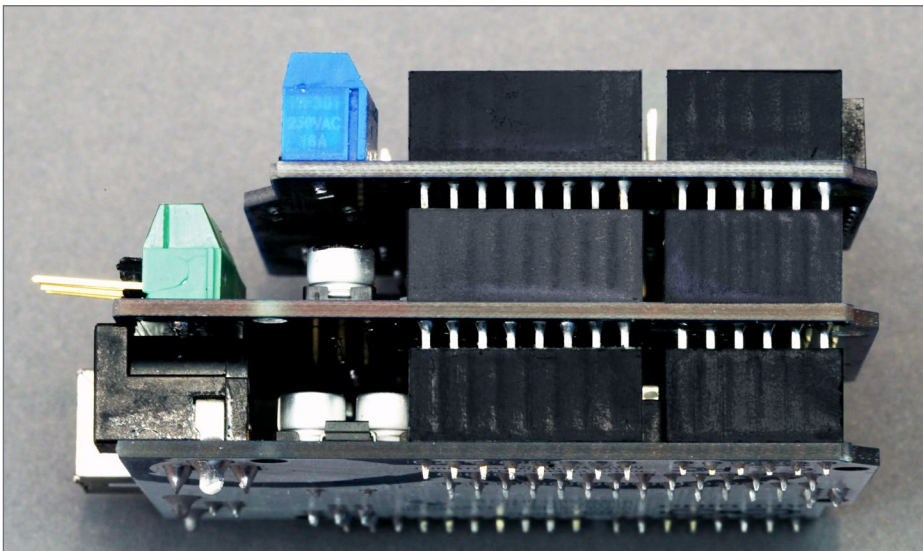
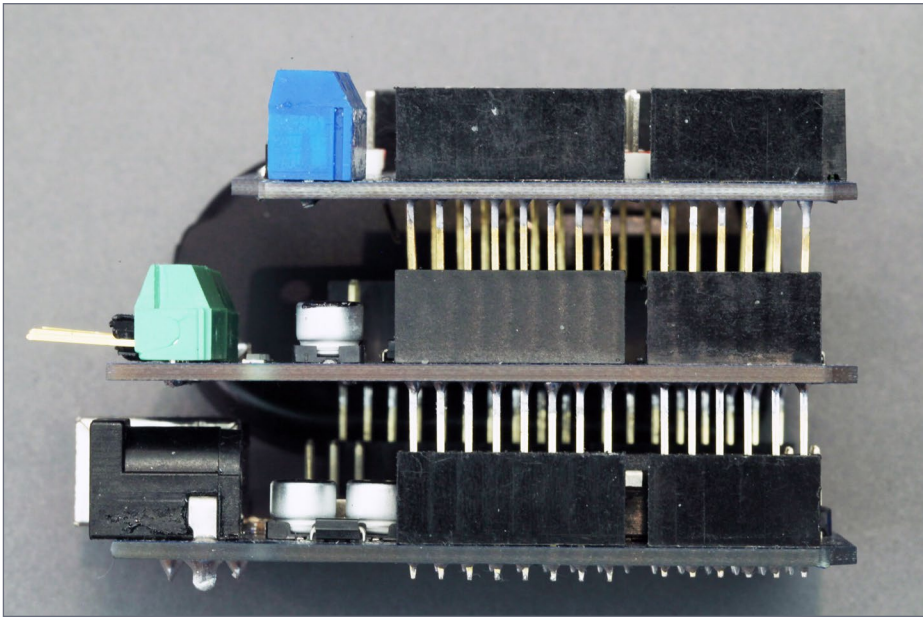
Sensor shield



• INDEX

• TABLE OF CONTENTS





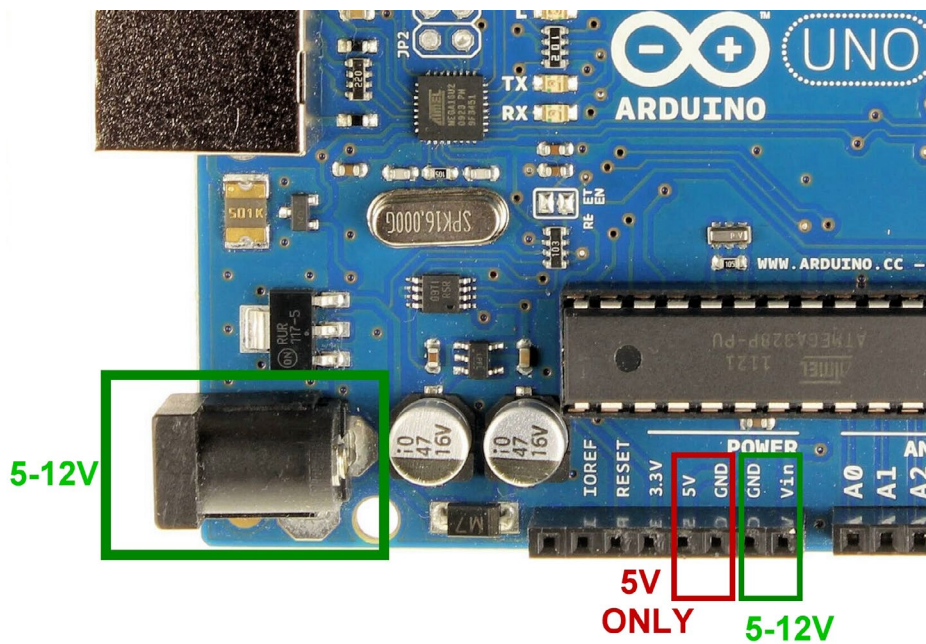
7. In this photo, you can see how the shield boards “stack” onto the Uno. In the top photo, I’ve aligned the pins, and in the bottom photo, I’ve pushed the boards all the way together. The Uno is on the bottom, the motor and sensor shields above.



• INDEX

• TABLE OF CONTENTS





8. Here are the different options for getting power to the Uno.

MRH has already published a guide to the Arduino **Pro Mini** in the November, 2014 issue, see: mrhpub.com/2014-11-nov/land/#99.

Powering your Arduino board

You can power the Uno with 5-12 volts DC power, including DC power supplies, batteries and small power adapters with a 2.5mm barrel connector (center positive with 5-12 Volts DC) either through the power socket or by feeding wires directly to the GND and VIN pin sockets on the Uno [8].

Or you can also feed 4.5-5.2 volts directly to the GND and 5V pin sockets on the Uno [8].

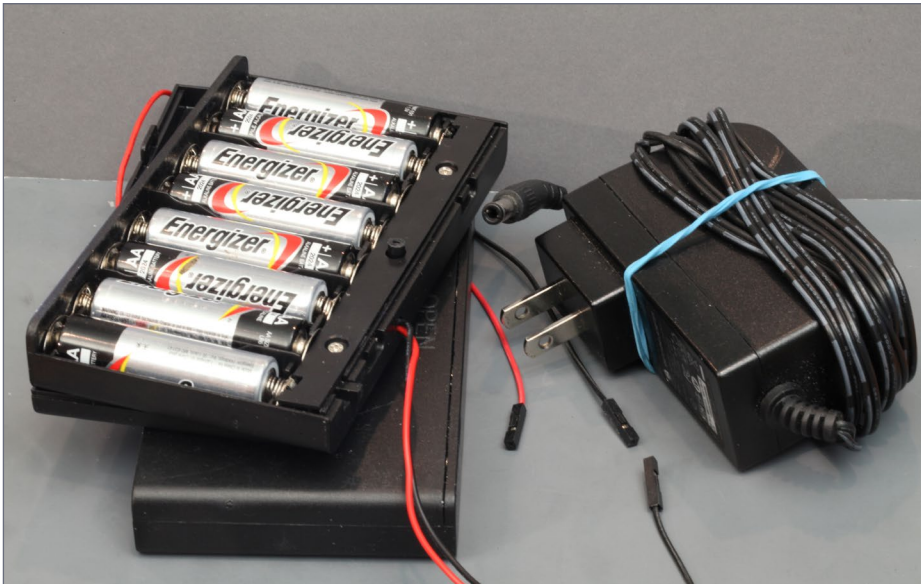
Finally, these boards can also get power just from the USB cable.



• INDEX

• TABLE OF CONTENTS





9. To source the power for your board, you can use either an eight AA battery pack (on left, 12V), or a 9V wall wart power adapter (on right) with an Arduino 5.5x2.5mm barrel plug.

Any of these power feed methods works, just pick one.

By itself an Arduino consumes very little power. I have built a Pro Mini controlled DCC board that operated an animation with a motor, lights and sound for more than 24 hours total just on eight AA batteries!

When it comes to the board's current limits (amperage), any one pin on the Uno can handle a max of 40 milliamps, with the total current through all the pins not exceeding a total of 200 milliamps (ma). However, keep in mind it's the total current actually being drawn that we're talking about, not the rated current.

To see how this works, read about a real-life example I did using eighteen LEDs in the sidebar: LEDs and Arduinos.



• INDEX

• TABLE OF CONTENTS



LEDs and Arduinos

An output pin of an Arduino can switch back and forth from 0.0-0.5 volts (LOW) to 4.5-5.0 volts (HIGH), and can handle currents up to 20ma per pin. However the total current handled by all pins must not exceed 200ma.

So if we want to light up eighteen LEDs at the same time [12], then we must limit the current to each LED down to 10 ma or lower, or light up fewer LEDs simultaneously. Fortunately, this is easy to do.

Each LED we use in these projects has its own “limiting” device called a resistor. These are quite cheap and readily available from many sources, including the ubiquitous Radio Shack.

Resistors are rated by a value of resistance stated in Ohms, and an associated power rating in Watts (w) – like a light bulb. Resistors in the range from 1,000 to 10,000 Ohms and any of either 1/4w, 1/8w, or 1/6w (available from China) will do fine here.

The higher the value (in Ohms) resistor the more dim the LED will glow. I recommend getting and keeping a range of values in your collection, like 470, 680, 1000, 2200, 4700, 5600, 6800, and 10,000. This way you can run tests to determine what an acceptable value will be for you to use in your electronic modeling projects.

LED manufacturers have dramatically improved the



10. LED examples.



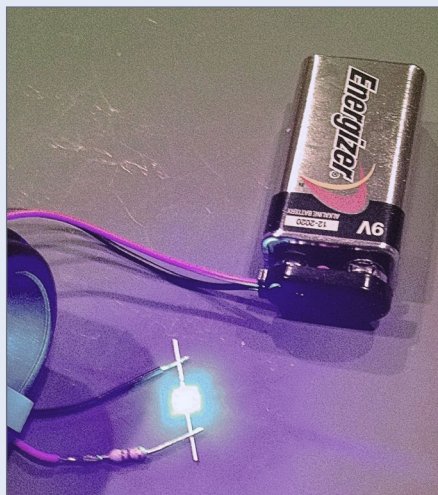
• INDEX

• TABLE OF CONTENTS



efficiency of LEDs over the last 20 years. The red LED pictured [10] was manufactured in 1981 and can barely be seen when powered with 18 ma. The small LED (second from the right) is so bright at 20 ma it hurts your eye to look into it for more than a fleeting glance.

The small green LEDs I use [11] light up very brightly at less than 0.2 ma. These are modern examples of what's currently being made.



11. This blue LED is being checked with a simple 9 volt battery and 470 Ohm resistor.

This efficiency difference is what is missing from most modeler discussions of LEDs and why I avoid formulas and equations in suggesting what resistor values to use.

Just keep a collection of various resistance values on hand and try several using the LEDs you will actually use in your project. Start with the highest value resistors (dimkest LED) and work down, until you get a pleasing brightness.

The vast majority of modelers' LEDs have metal leads similar to those pictured. Note that four of the five LEDs have a long and a shorter metal lead. The longer lead goes to the positive voltage if you want the LED to light up. The shorter lead is the negative side and is usually connected to GND or Ground (as labeled on the Arduino).

In additional reading, you may find the positive LED lead called the "anode" and the negative LED lead called the "cathode."



Also note the red LED on the left [10] has 2 leads of the same length. There is an easy way to identify the positive and negative leads on this type of LED – and test it at the same time! Get a 9 volt battery and a battery connector (Radio Shack #2700325) along with a 470 Ohm $\frac{1}{4}$ Watt resistor (Radio Shack #2711317).

Connect one side of the resistor to the Red wire of the 9 volt battery connector (solder it or just twist it on). Touch the free end of the resistor to one leg of the ‘unknown’ LED and touch the black lead from the battery connector to the other leg of the LED. If the LED does not light, reverse the connections to the LED.

When the LED lights, you will know that the black lead is connected to the negative side of the LED. It will also indicate whether or not you have a working LED. For the vast majority of LEDs you will use in modeling, this test will work fine. ■



11. I wired all 18 LEDs shown here with 10k resistors, resulting in a total combined current draw for all the LEDs of less than 20ma – well below the Uno's 200ma board limit.



• INDEX

• TABLE OF CONTENTS



Let's look at some projects

Let's look at a range of easy projects that can enhance your layout. I have divided them into lighting control, servo control, and sound generation. These projects do not require any complex computer interfacing but are simple stand-alone projects using just the Arduino and some electronic components, which is why I chose them.

Controlling these projects: Okay, let's say you build one of these projects. So where is the on/off button?

For most of these projects, the "on/off button" is pin 14 (A0). On the Arduino, this is a master control pin that when set LOW (connected to Ground or dropped to zero Volts) turns off the project's function(s). When set to HIGH (a voltage greater than 3.5 volts or more), the board turns on.

One fun way to control this pin is with a special sensor that senses heat using infrared light called a PIR Sensor [12].

After setting up a project, you make three connections to the PIR sensor. Make one connection to 5 Volts, another to Ground, and finally to pin 14 (the master control pin) and point the sensor away from you.

When you put your hand or body in the field of view of the PIR sensor, the project you are controlling will turn on. When you move out of view to the side, the project continues to function briefly and then stops.

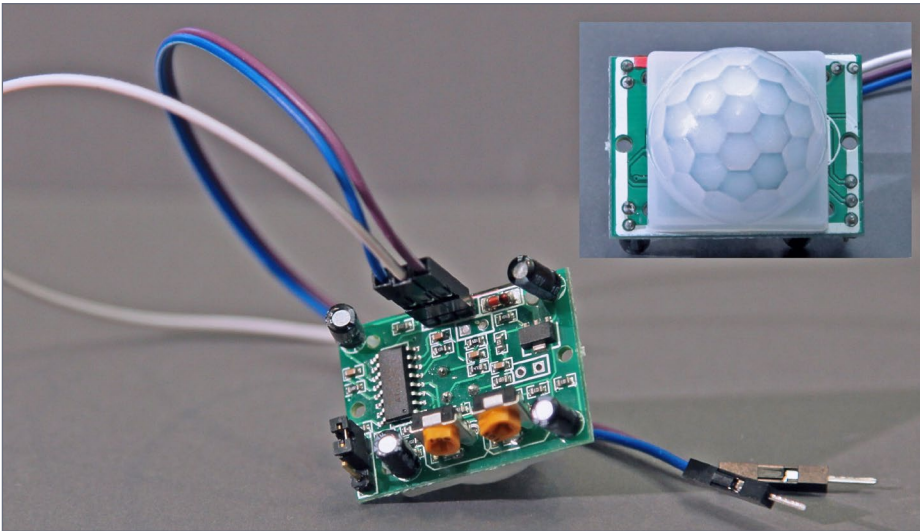
If you put a piece of wood, cardboard or plastic in front of the PIR sensor – it will not respond! It's only activated by your body heat. If you want to create an animation or turn something on



• INDEX

• TABLE OF CONTENTS





12. Here is a PIR sensor. Use the jumper on the lower left to set to retrigger each time, and adjust the yellow variable resistors to set the time delay and sensitivity.

when a person is in front of your model, put this sensor where it can “see” them come by. How cool is that?

This is great for clubs and public shows as an “attention getter.” You do need to learn how to set the PIR sensor to “retriggerable” (done with a jumper), and then make an adjustment to the two yellow variable resistors [12] to set the “time on delay” and the “sensitivity.”

Project setup

The following projects all list the sketch file which needs to be loaded onto your Arduino (see the subscriber bonus downloads). I describe the sketch loading process a special how-to supplement in [this issue's bonus downloads](#).



• INDEX

• TABLE OF CONTENTS



Projects that include the control notation “Control: pin 14 (A0)” indicate using control pin 14 (A0) as the master control pin for turning the function on and off. If you just leave this pin unconnected, then the project starts to function as soon as you apply power to the Arduino board.

The “Setup:” reference in the project points to the specific pictured pin connections needed.

In the project descriptions I highlight key settings (with names used in the sketch) that affect operation and results you see and/or hear. You can use these as is, with no modification. If you like, you can easily edit them and experiment to see what you might prefer for your situation.

Whatever changes you make can be saved for future use. You do not need to understand programming to use any of these, but you can learn if you want!

I am not a proponent of asking modelers to learn how to program – all you need to do is learn how to use this new IDE app – and if you can use a word editor you can use this app. You can take using the IDE as far as you want, but I will say learning to use the IDE can add a great deal to your modeling enjoyment!

In the projects, I use a notation for led_pins to show which Uno or Pro Mini pins to make the appropriate connections. I use pictures rather than diagrams to give you setup details.

To get the sketches, see this issue’s subscriber bonuses. A step-by-step procedure showing how to setup and load a sketch into your Arduino is [in this issue’s bonus downloads](#).

These projects give you some idea of the capability and versatility of the Arduino for modeling. And they’re just a start!



• INDEX

• TABLE OF CONTENTS



Once you decide which project you would like to use:

1. Set up IDE editor on your computer (see [bonus downloads](#)).
2. From the bonus downloads this issue, copy all the files in the **MRH** folder to your Arduino sketches folder – usually:
...\[Documents\Arduino](#)\ on a Windows machine.
3. From the bonus downloads, copy all the files in **MRH_libraries** folder to your Arduino libraries folder – usually:
...\[Documents\Arduino\libraries](#)\ on a Windows machine.
4. Wire your Arduino according to the pictures or diagrams.
5. Download the specific sketch to your Arduino using the instructions in the [bonus downloads](#) and enjoy!

Lighting projects

Watch the video below to see what the lighting projects do.



Playback problems? [Click here ...](#)



• [INDEX](#)

• [TABLE OF CONTENTS](#)



BARRIER DIRECTION LIGHTS

Sketch folder: Barrier_Lights

Setup: Lights in a row [photo 14]

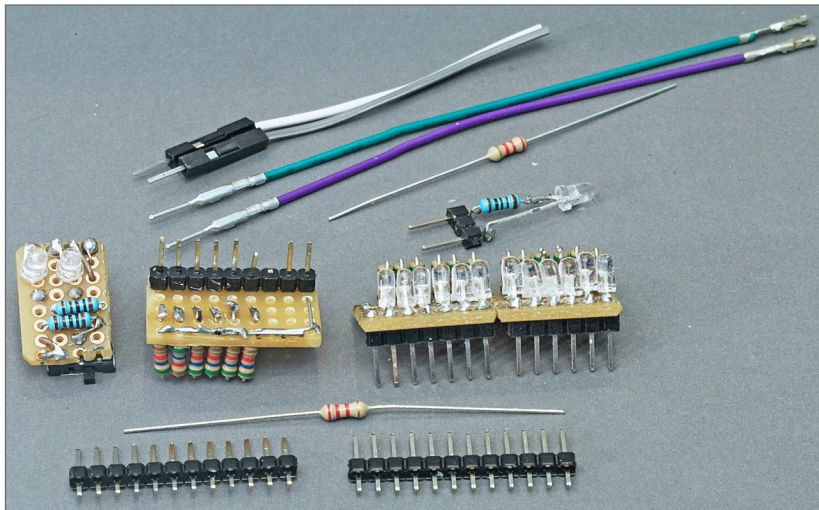
Control: pin 14 (A0)

Connected pins: led_pins - 2,3,4,5,6,7,8,9,10,11,12,13

This project blinks a row of LEDs from the middle out with the timing determined by sketch setting `delta`. The sketch defaults to 300 for `delta`.

For each LED, you can connect the positive leg of the LED to a resistor and then insert the resistor end into any Uno pin. The negative end of the LED needs to connect to Ground. Since Ground is literally the common wire, then all the LED negative connections can be wired together and only a single wire would run back to Uno Ground [13].

I mounted the LEDs and resistors in sets of six with header pins on a small piece of perfboard (a.co/hWSnKzQ), see below.



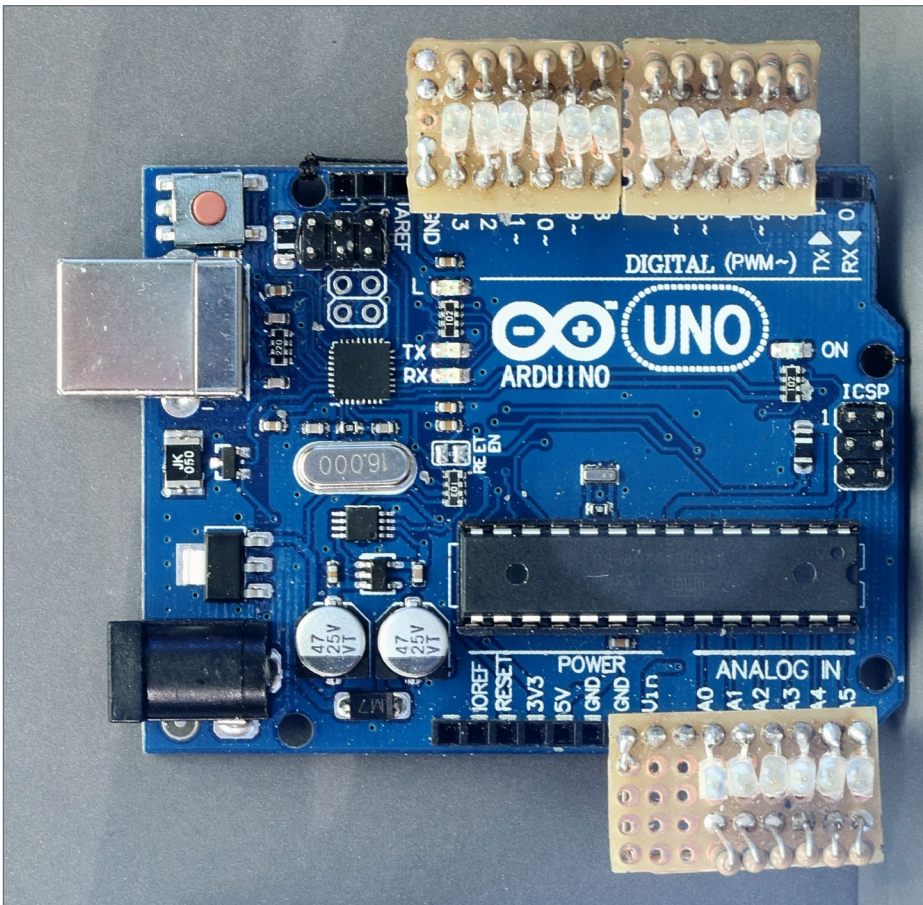
13. Here are the LED wiring components used for many of the lighting projects.




• INDEX

• TABLE OF CONTENTS





14. Lights in a row setup.

 [Click here for reader comments](#)



• INDEX

• TABLE OF CONTENTS



BLINK SINGLE LED A NUMBER OF TIMES

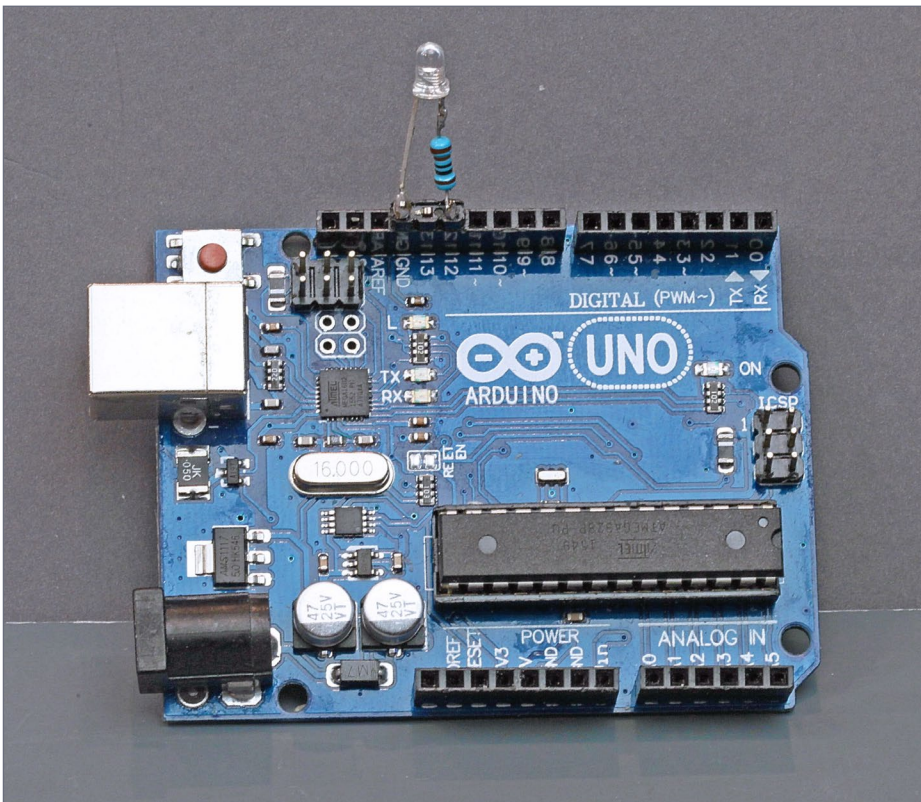
Sketch folder: Blink_Single

Setup: Single LED [photo 15]

Control: pin 14 (A0)

Connected pins: led_pin - 12

This project repeatedly blinks a single LED a `blink_number` number of times with the timing determined by sketch setting `delta`. The sketch defaults to 6 for `blink_number` and 95 for `delta`.



15. Single LED setup.



• INDEX

• TABLE OF CONTENTS



RANDOM BUILDING LIGHTING

Sketch folder: Building_Lights

Setup: Lights in a row [photo 14]

Control: pin 14 (A0)

Connected pins: led_pins - 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15 (A1), 16 (A2), 17 (A3), 18 (A4), 19 (A5)

This project randomly blinks 17 LEDs in a random pattern allowing a 60% "ON" time. Timing determined by sketch setting `tim_delay`, which defaults to 1100. People enter a room, turn on lights, stay for some time, and then leave turning off the lights. This sketch tries to account for such behavior with a pseudo-randomness that looks convincing to the casual observer.



16. Random building lights in action on the layout.



• INDEX

• TABLE OF CONTENTS



MONOCHROME TELEVISION

Sketch folder: BW_TV

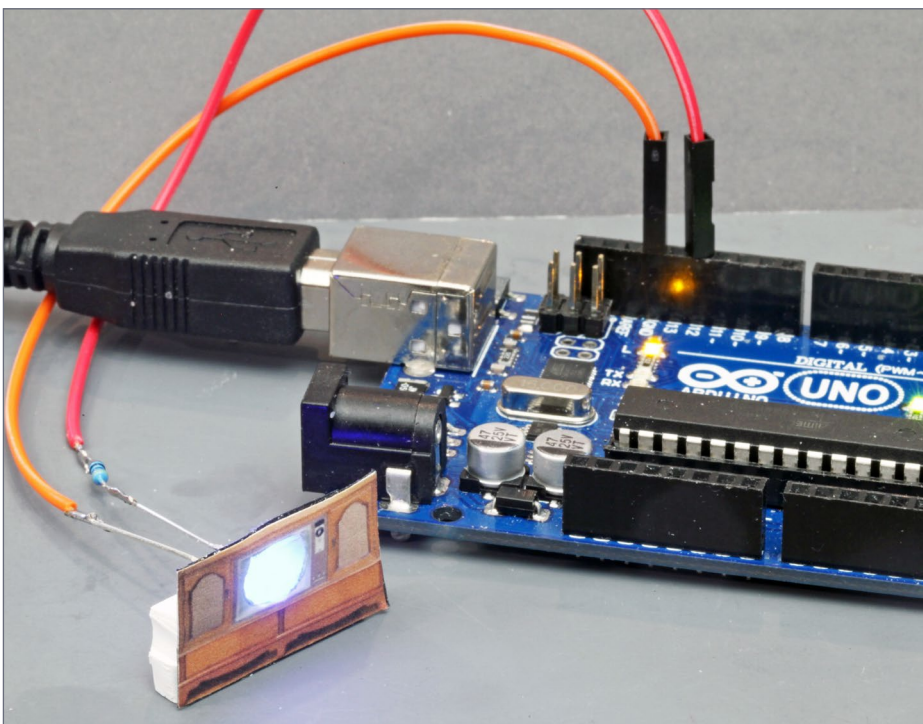
Setup: Monochrome TV [photo 17]

Control: pin 14 (A0)

Connected pins: TV - 12

This project randomly blinks a blue LED in a random pattern simulating a flickering monochrome TV screen. Timing determined by sketch setting `change_delay`, which defaults to 100.

I used was a 3mm white LED that was tinted blue with a permanent marker, and placed it about ¼ inch back from a paper screen I pasted to a scaled down TV cabinet picture.



17. Monochrome TV setup.



• INDEX

• TABLE OF CONTENTS



CHASE PATTERN FOR THEATER MARQUEE

Sketch folder: Chase_Lights

Setup: Lights in a row [photo 14]

Control: pin 14 (A0)

Connected pins: led_pins - 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15 (A1), 16 (A2), 17 (A3), 18 (A4), 19 (A5)

This project blinks a row of LEDs with a specific pattern defined by sketch setting `chase_pattern`. Default pattern is: 0,1,1,1,1,1,0,1,1,1,1,0,1,1,1.

Timing determined by sketch setting `delta`, which defaults to 46.

CHASE PATTERN-2 FOR THEATER MARQUEE

Sketch folder: Chase_Lights2

Setup: Lights in a row [photo 14]

Control: pin 14 (A0)

Connected pins: led_pins - 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15 (A1), 16 (A2), 17 (A3), 18 (A4), 19 (A5)

This project blinks a row of LEDs with a specific pattern defined by sketch setting `chase_pattern`. Default pattern is: 0,1,1,1,0,1,1,1,0,1,1,1,0,1,1,0.

Timing determined by sketch setting `delta`, which defaults to 76.

This uses a different pattern and different timing to demonstrate what else can be done.



• INDEX

• TABLE OF CONTENTS



CHRISTMAS TREE

Sketch folder: Christmas_Lights

Setup: Lights in a row [photo 14]

Control: pin 14 (A0)

Connected pins: led_pins - 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15 (A1), 16 (A2), 17 (A3), 18 (A4), 19 (A5)

This project randomly blinks 17 LEDs in a true random pattern. Timing determined by sketch setting `delta`, which defaults to 300.

This is an example of using the same sketch developed on the Uno, but loaded into an Arduino Pro Mini. Look closely: the Christmas tree is actually mounted on the Pro Mini in the photo below [17].

I used a small bristle brush tree, added some beads, glitter, and “snow” from Woodland Scenics. I used white LEDs colored with permanent ink markers, Tamiya clear acrylics, dyes, and so on.



18. Christmas tree project on the layout (somewhat whimsically.)



• INDEX

• TABLE OF CONTENTS



COLOR TELEVISION

Sketch folder: Color_TV

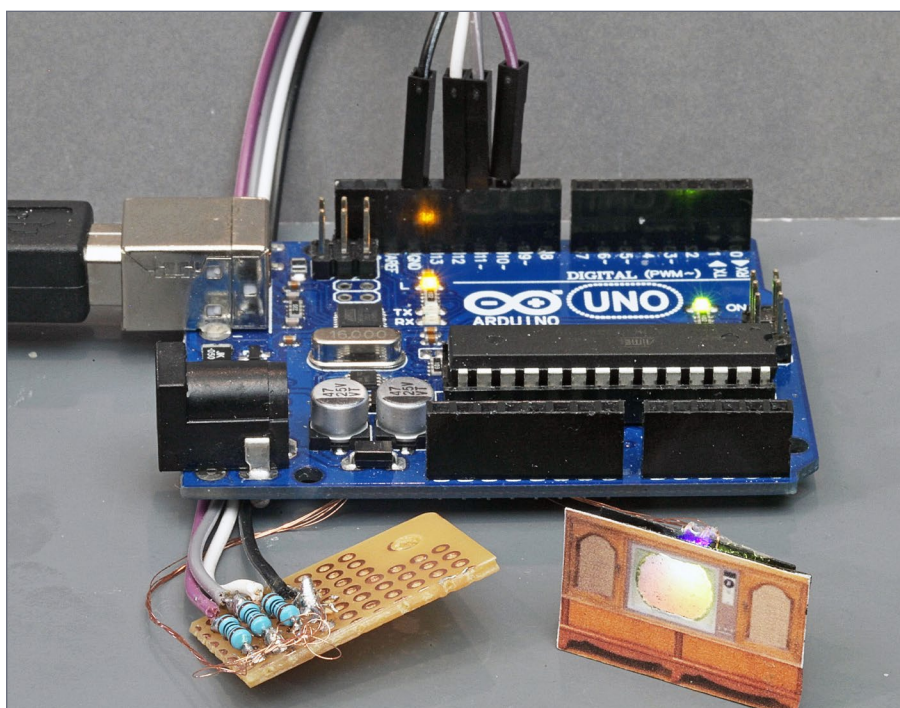
Setup: Color TV [photo 18]

Control: pin 14 (A0)

Connected pins: TVRED - 12, TVGRN - 11, TVBLU - 10

This project randomly blinks red, blue and green LEDs in a weighted random pattern simulating a flickering color TV screen. Timing is determined by sketch setting `change_delay`, which defaults to 150.

I used 0602 prewired LEDs colored red, green, and blue set back ¼ inch from a paper screen glued to a cutout scale TV paper picture. I placed the resistors on a small perf board as a connection strip for the LED connections, and held the jumpers to the Uno.



19. Color TV setup.



• INDEX

• TABLE OF CONTENTS



FIRE LIGHT

Sketch folder: Fire_Light

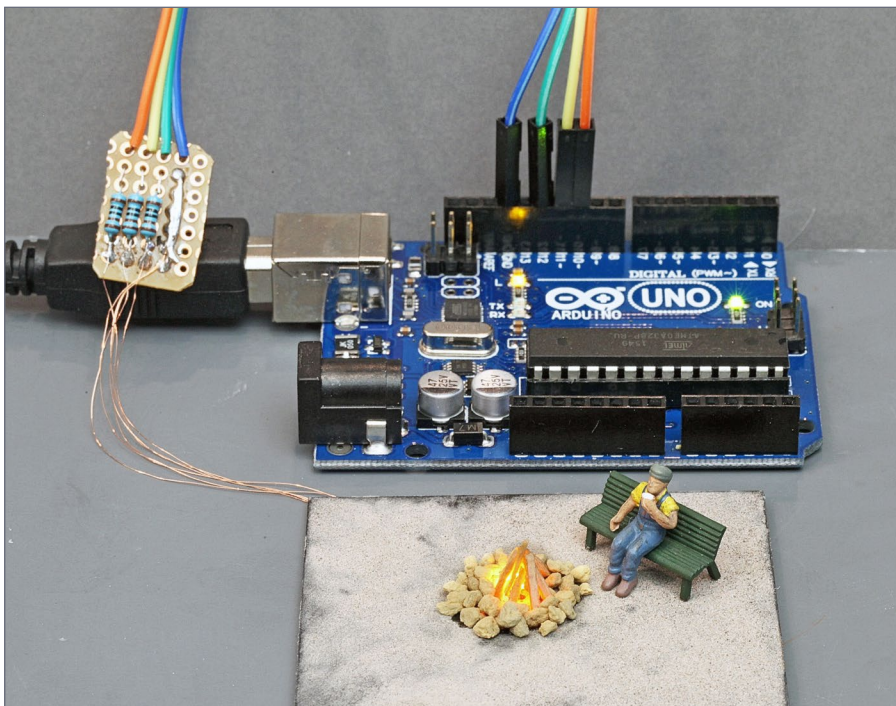
Setup: Fire light [photo 19]

Control: pin 14 (A0)

Connected pins: TVRED - 12, TVYEL - 11, TVWHT - 10

This project randomly blinks red, yellow and white LEDs in a weighted random pattern simulating a flickering fire. Timing is determined by sketch setting `change_delay`, which defaults to 100.

This is actually similar to the color TV. Here the I colored the 0602 LEDs red, yellow, and white. The timing is different for each so the order is important – the white is fired intermittently as an exploding coal. The LEDs stick 1/8 inch up into the wood pile of the “fire.”



20. Fire light setup.



• INDEX

• TABLE OF CONTENTS



ROTATING BEACON OR SEARCH LIGHT

Sketch folder: Rotating_Beacons

Setup: Lights in a row [photo 14] or Single LED [photo 15]

Control: pin 14 (A0)

Connected pins: led_pin1 - 12 and led_pin2 - 10

This project repeatedly blinks 2 LEDs with a gradual fade on and fade off. Timing is determined by sketch settings `deltatime1` (default 700) and `deltatime2` (default 300).

ADVERTISEMENT

Make LocoNet wiring extremely simple!



Suitcases belong in your baggage cars,
not under you layout!

Introducing the DCC Vampire II ...
this guy does the job of two suitcase
connectors!
And really holds the connection good
too!



CLICK NOW TO DISCOVER THE AMAZING VALUE!

Dealer Inquiries Accepted



• INDEX

• TABLE OF CONTENTS



RAILROAD CROSSING LIGHTS

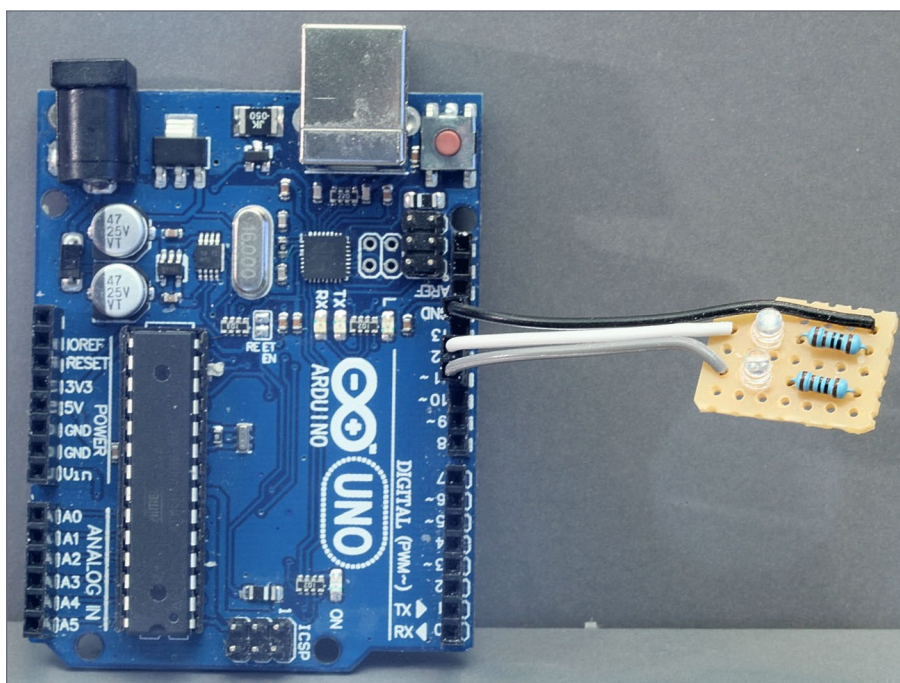
Sketch folder: RR_Crossing_Lights

Setup: Dual LED connections [photo 20]

Control: pin 14 (A0)

Connected pins: led_pin1 - 11 and led_pin2 - 12

This project alternately blinks 2 LEDs . Flash timing is determined by sketch setting `delta`, which defaults to 630.



21. Dual LED connections setup.



• INDEX

• TABLE OF CONTENTS



AIRPORT RUNWAY LIGHTS

Sketch folder: Runway_Lights

Setup: Lights in a row [photo 14]

Control: pin 14 (A0)

Connected pins: led_pins - 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15 (A1), 16 (A2), 17 (A3), 18 (A4), 19 (A5)

This project blinks a row of LEDs from start to finish with one LED on at a time like airport runway lights. Timing determined by sketch setting `delta`, which defaults to 46.

STROBE LIGHTS

Sketch folder: Strobes

Setup: Lights in a row [photo 14]

Control: pin 14 (A0)

Connected pins: led_pin1 - 11 and led_pin2 - 12

This project repeatedly blinks 2 LEDs simulating 2 strobe lights.

Timing determined by sketch settings `ON_TIME1` & `OFF_TIME1` as well as `ON_TIME2` & `OFF_TIME2` respectively.

WELDER IN ACTION (INDIRECT VIEW)

Sketch folder: Welder_Indirect

Setup: Dual LED connections [photo 20]

Control: pin 14 (A0)

Connected pins: welderpinWH-12 and welderpinBL-11

This project controls two white LEDs: one tinted blue with a marker, to simulate a welding operation with the material being welded not in direct view. By the way, Tamiya clear acrylics also work well for color-tinting white LEDs.



• INDEX

• TABLE OF CONTENTS



Servo Motor projects

Watch the video below to see what the servo motor projects do.



Playback problems? [Click here ...](#)

A servo is a motor-driven rotary mechanism that is electronically controlled to allow precise angular positioning and speed control.

Rather than get mired in a lot of details about servos here, let me simply point you at the wealth of info on the MRH site in Peter Randerson's *Servos 101* thread: mrhmag.com/node/25958.

Low-cost servos are available from many radio control (R/C) hobby sources. Besides driving LEDs, Arduinos can easily drive servo motors directly from each pin.

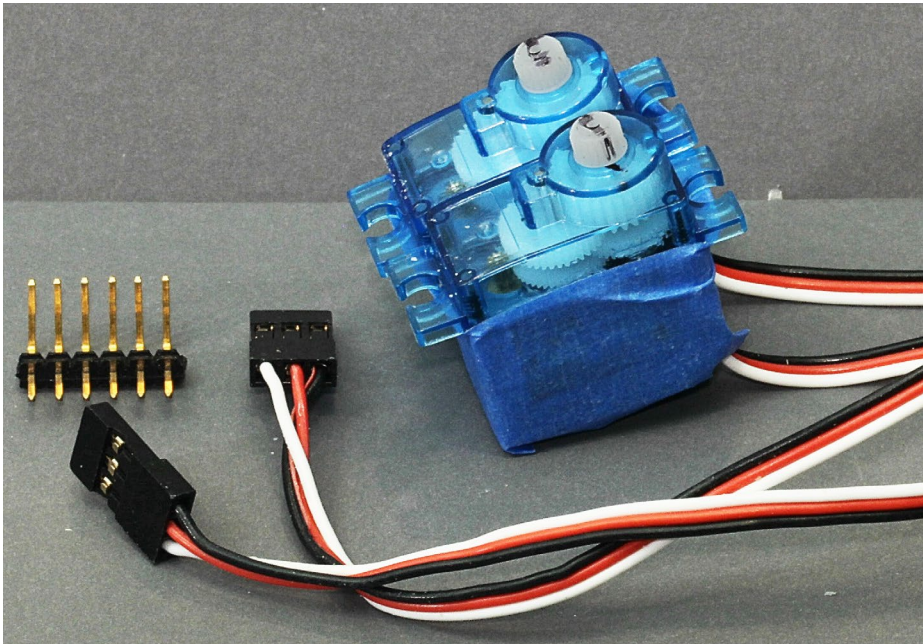
What may not be obvious is that while a single servo motor can be controlled with one Uno pin, it is not a simple on/off switch. Rather, the Uno is switching the servo control pin on and off about 50 times a second!



• INDEX

• TABLE OF CONTENTS





22. Two 9G sub-micro sized servos.

This is just a bit faster than most of us can toggle a panel switch! The Arduino also has to control the “on time” of the switch to consistently set the servo position.

For these projects, we install a Sensor Shield into the Arduino Uno board pins to make a “stack”. The Sensor shield has conveniently replicated the Arduino pins sockets side by side with 5V and Ground pin sockets – perfect for operating servos.

Be aware these servo projects get a bit more involved than the LED examples, especially as we work our way through the project examples. We delve more into editing the sketch files, so you will want to check out the last section of this article on using the Arduino IDE to edit a sketch file.

Let's see what we can do with some low cost servos and an Arduino stack with a Sensor Shield!



• INDEX

• TABLE OF CONTENTS



SERVO MOVING END TO END CONTINUOUSLY

Sketch folder: Servo_Back_and_Forth

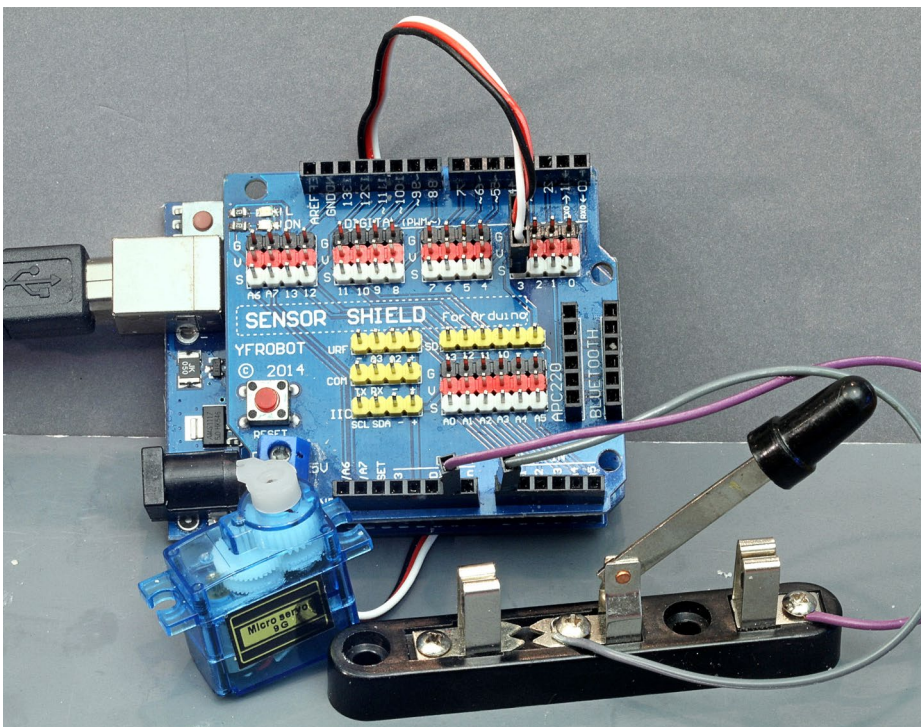
Setup: Single servo with switch [photo 22]

Control: pin 14 (A0)

Connected pins: servo_pin - 3

This project moves the servo arm back and forth repeatedly. The starting point is determined by `servo_start` (default 20) while `servo_stop` (default 168) sets the stopping point.

This motion can move a pump, figures, and other items with a repetitive motion. A servo needs +5 Volts (red wire), ground (black wire), and control (usually a white or yellow wire) which connects through the Sensor Shield to an Uno pin.



23. Single servo with switch.



• INDEX

• TABLE OF CONTENTS



SERVO MOVING SLOWLY END TO END CONTINUOUSLY

Sketch folder: Servo_Slow_Back_and_Forth

Setup: Single servo with switch [photo 22]

Control: pin 14 (A0)

Connected pins: servo_pin - 3

This project moves the servo arm slowly back and forth repeatedly. The movement speed is set by `servo_delay` (default 5) while the end points are set by `servo_start` (default 20) and `servo_stop` (default 168).

ONE SERVO CONTROLLED BY SIMPLE SWITCH

Sketch folder: Servo_One_Switched

Setup: Single servo with switch [photo 22]

Control: pin 14 (A0)

Connected pins: servo_pin - 3

This project moves the servo arm from end to end, triggered by a simple on/off switch connected to pin 14 (A0). The starting point is determined by `servo_start` (default 20) while `servo_stop` (default 168) sets the stopping point.

You need one Uno pin for the servo control and one Uno pin to attach the switch. This is a simple on/off switch with two connections, one to Uno pin 14 (A0) and the other connected to Ground. Any kind of simple switch will work.

If you look at this sketch in the IDE editor, lines 8 and 9 say:

```
#define servo_start 20      // Servo start position
#define servo_stop  168     // Servo stop position
```

If you change either of these numbers, you will change the traverse of the servo arm. These represent positions from 0 to 180 degrees in an arc. For cheap generic 9G servos, you will likely not be able to go completely from 0 to 180 degrees.



• INDEX

• TABLE OF CONTENTS



6 PUSHBUTTON CONTROLLED SERVOS WITH LED INDICATOR

Sketch folder: Switched_6_Servos

Setup: Servo setup 2 [photo 24, next page]

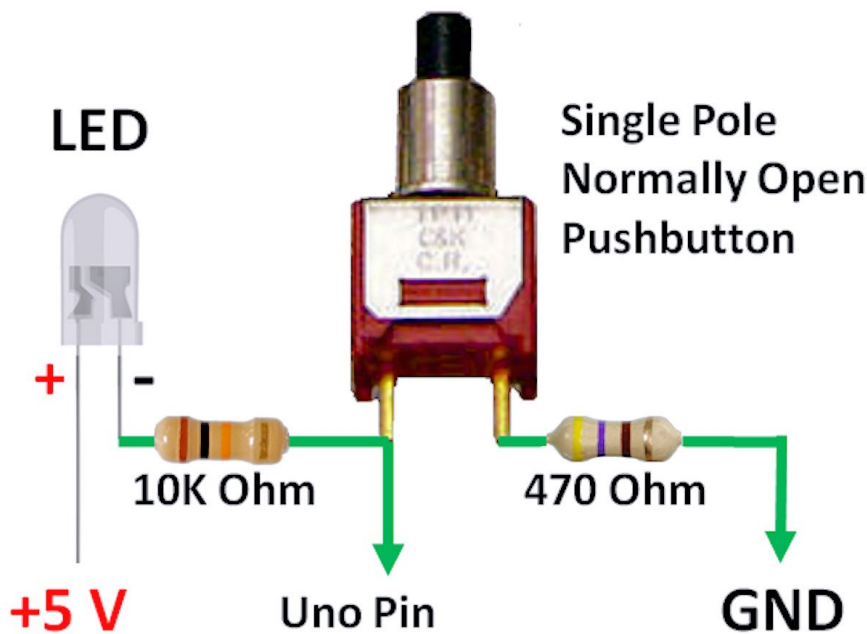
Control: pin 14 (A0)

Connected pins: servos - spins - 2, 3, 4, 5, 6, 7;

pushbuttons - pins - 14 (A0), 15 (A1), 16 (A2), 17 (A3), 18 (A4), 19 (A5)

This project controls 6 servos with corresponding start and stop positions using push buttons. The starting positions are determined by `sstart-25,25,25,25,25,25` and the stop positions by `sstop-160,160,160,160,160,160`.

Each push of the corresponding button causes the servo to seek the opposite position and changes the state of the corresponding LED, using the pushbutton/LED connections shown below [23].



24. Push button and LED wiring diagram.



• INDEX

• TABLE OF CONTENTS



8 PUSHBUTTON CONTROLLED SERVOS WITH LED INDICATOR

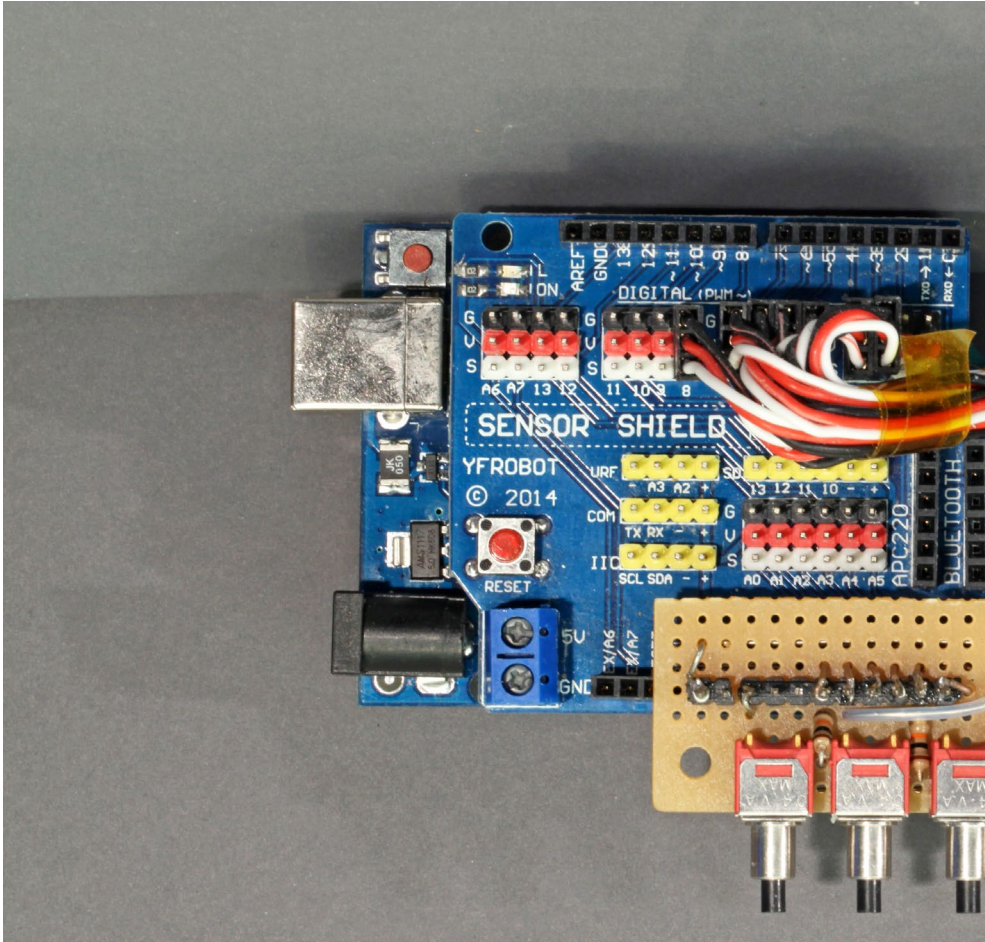
Sketch folder: Switched_8_Servos

Setup: Servo setup 2 with eight servos and buttons [photo 24]

Control: pin 14 (A0)

Connected pins: servos - spins - 2, 3, 4, 5, 6, 7, 8, 9;

pushbuttons - pins - 11, 12, 14 (A0), 15 (A1), 16 (A2), 17 (A3), 18 (A4), 19 (A5)



25. Servo setup 2, using an Uno and a Sensor Shield.



• INDEX

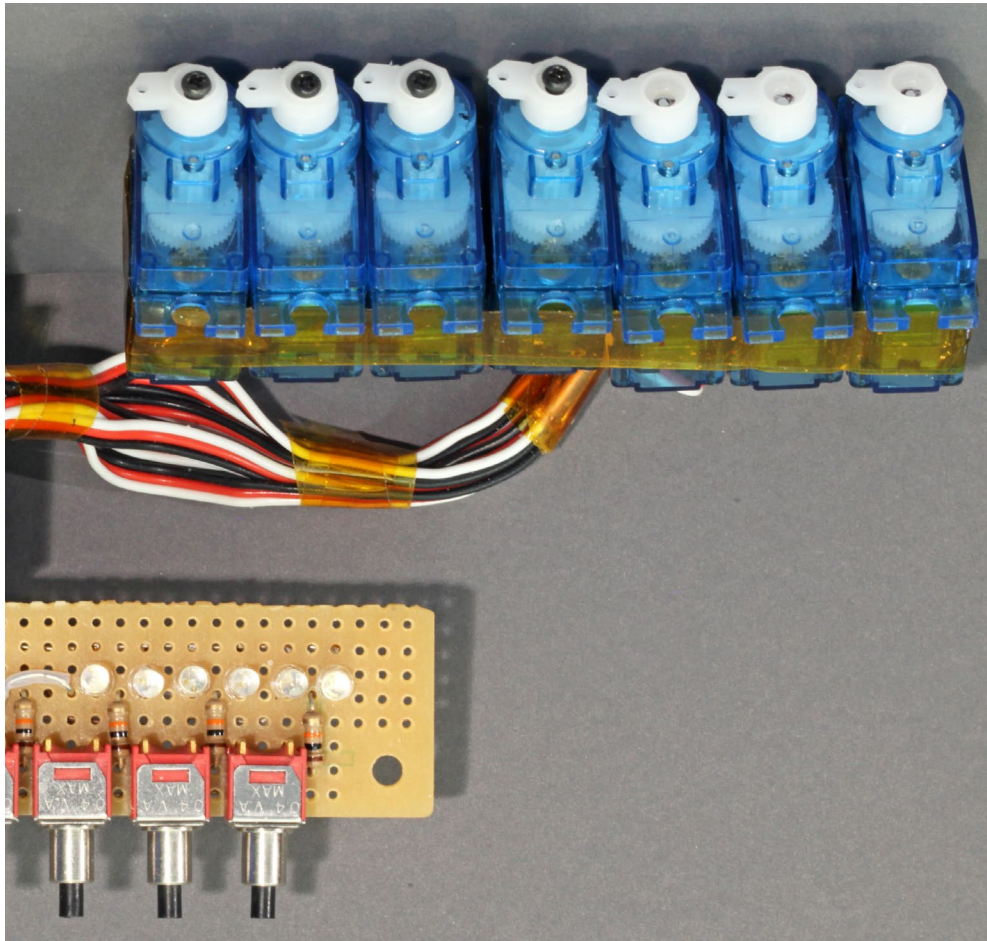
• TABLE OF CONTENTS



MODELER'S INTRO TO THE ARDUINO | 38

This project controls 8 servos with corresponding start and stop positions using push buttons. The starting positions are determined by `sstart-25,25,25,25,25,25,25,25` and the stop positions by `sstop-160,160,160,160,160,160,160,160`.

Each push of the corresponding button causes the servo to seek the opposite position and changes the state of the corresponding LED, using the pushbutton/LED connections pictured [23].



• INDEX

• TABLE OF CONTENTS



8 PUSHBUTTON CONTROLLED ROUTES USING SERVOS WITH LED INDICATOR

Sketch folder: Switched_8_Routes

Setup: Servo setup 2 with eight servos and buttons [photo 24]

Control: pin 14 (A0)

Connected pins: servos - spins - 2, 3, 4, 5, 6, 7, 8, 9;
pushbuttons - pins - 11, 12, 14 (A0), 15 (A1), 16 (A2), 17 (A3), 18 (A4), 19 (A5)

This project controls 8 routes. A route is simply a defined set of track switch positions, often in a yard, that collectively defines a single path through the yard. To form this route, the 8 servos each need to be set to a given start or stop position (think turnouts with two directions).

Each push of a corresponding button sets a route. A route is a collection of servo settings in this table found in this project's sketch:

```
routes [8][8] = { // ROUTE TABLE Definition for each route & servo
  {0,0,1,1,0,0,0,0}, // route 0 pushbutton 0
  {0,0,0,0,1,1,0,0}, // route 1 pushbutton 1
  {0,0,0,0,0,0,0,0}, // route 2 pushbutton 2
  {1,1,1,1,1,1,1,1}, // route 3 pushbutton 3
  {1,1,1,0,0,0,0,0}, // route 4 pushbutton 4
  {0,0,0,1,1,1,0,0}, // route 5 pushbutton 5
  {0,0,0,0,0,1,1,1}, // route 6 pushbutton 6
  {1,0,1,0,1,0,1,0} // route 7 pushbutton 7
```

This is a useful way to throw track switches controlled by servos. For each route 0-7, a "0" in the routes table row sets the corresponding servo to its saved **sstart** position and a "1" sets the corresponding servo to its saved **sstop** position. It also changes the state of the corresponding route LED.

You can set the routes (each table row) up any way you want!



• INDEX

• TABLE OF CONTENTS

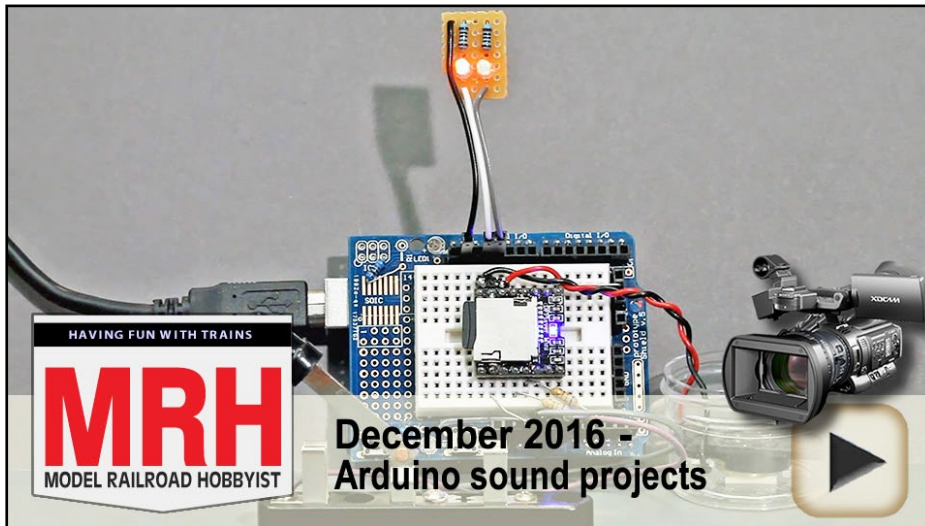


MODELER'S INTRO TO THE ARDUINO | 40

Final note: For all the 6 and 8 servo pushbutton-controlled projects, the 5 volt power to the servos should be a *separate power source*. Neither the Uno nor the Pro Mini can provide enough power via the on board regulator. Low cost 9G servos can have a peak current draw of over 1 amp!

Sound projects

Watch the video below to see what the sound projects do.



Playback problems? [Click here ...](#)

Note: All the sound projects use a small, low-cost module called a DFPlayer. This is available from [dfrobot.com \(dfrobot.com/wiki/index.php/DFPlayer_Mini_SKU:DFR0299\)](http://dfrobot.com/wiki/index.php/DFPlayer_Mini_SKU:DFR0299) or from eBay.com (ebay.com/itm/191947254444). You can also Google for DF-Player and find still more sources.

More detailed documentation on the DFPlayer is included in this month's [subscriber bonus downloads](#).



• INDEX

• TABLE OF CONTENTS



CROSSING LIGHTS & BELL SYNCHRONIZED TOGETHER

Sketch folder: RR_Crossing_Bell_Synched

Setup: Sound setup 1 [photo 25]

Control: pin 14 (A0)

Connected pins: led_pin1-11, led_pin2-12, bell_pin-15

This project alternates the flashing of two LEDs synchronized with the “gong” sound of the crossing bell. This an older style of prototype crossing signal. Flash rate depends on setting **delta**, which defaults to 540.

This project is a good example illustrating simple synchronization of each crossing lamp with the single strike of the bell. Using an Arduino, the lights and sound can all be synchronized to a very good degree of accuracy to convincingly simulate such older crossing signals.

CROSSING LIGHTS & BELL FREE RUNNING

Sketch folder: RR_Crossing_Bell_FreeRun_PB

Setup: Sound setup 1 [photo 25]

Control: pin 14 (A0)

Connected pins: led_pin1-11, led_pin2-12, bell_pin-15 (A1), silence_pin-16 (A2), control_pin-14 (A0)

This project alternates the flashing of two LEDs. Flash rate depends on setting **delta**, which defaults to 745.

A pushbutton is connected to control_pin- 14 (A0). Each press switches the project on – off – on – etc.

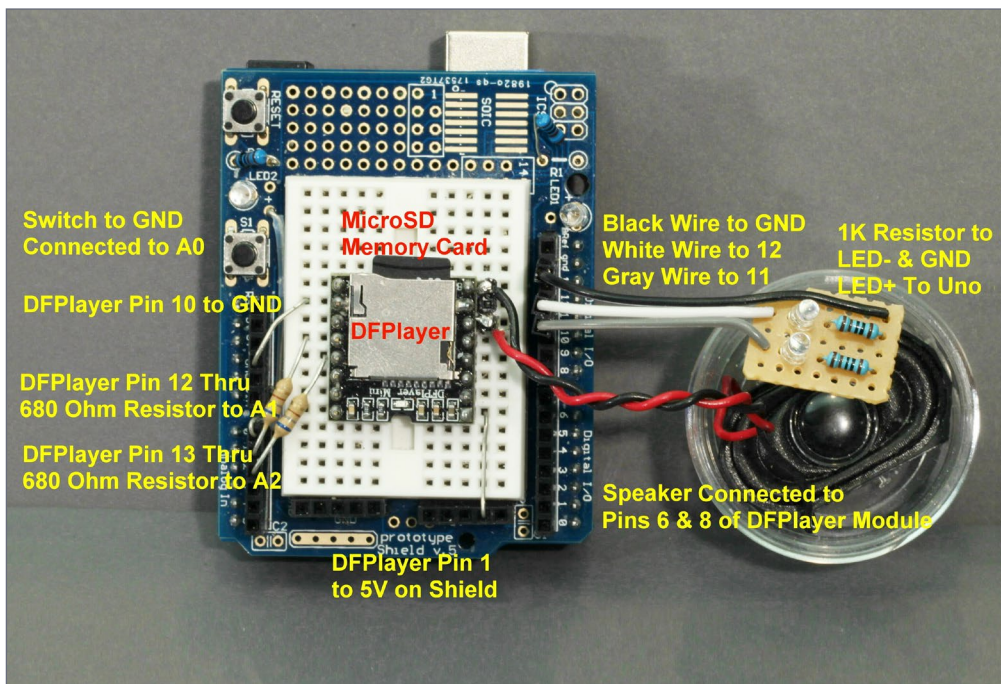
The crossing bell runs at an independent rate, determined in the prototype by the bell mechanism itself. This is a more modern style of prototype crossing signal.



• INDEX

• TABLE OF CONTENTS





26. Sound setup 1: Uno and DFPlayer module connections on a Prototype Shield using a small speaker.

The speaker [25] is a small oval speaker that I obtained from a discarded computer laptop although you can find a very similar speaker on Amazon at this link: a.co/3TkbYbp. I placed the speaker in an open plastic box about 2 inches deep to improve its sound.

The resistors in series with the LEDs are 1000 Ohm.

The sound files are included in the [subscriber bonus downloads](#), and they must be loaded onto a micro SD memory card for use with the DFPlayer [25].

You can get a USB to SD card reader and a micro SD Card to standard SD Card adapter, then load the files onto the micro SD card from your computer [26].



• INDEX

• TABLE OF CONTENTS



CROSSING LIGHTS & BELL FREE RUNNING ACTIVE HIGH

Sketch folder: RR_Crossing_Bell_SW_HIGH

Setup: Sound setup 1 [photo 25]

Control: pin 14 (A0)

Connected pins: led_pin1-11, led_pin2-12, bell_pin-15 (A1), silence_pin-16 (A2), control_pin-14 (A0)

This project alternates the flashing of two LEDs. Flash rate depends on setting **delta**, which defaults to 745.

A simple on/off switch is connected to control_pin- 14 (A0). The project switches ON when the control_pin is HIGH.

The crossing bell runs at an independent rate, simulating a more modern style of prototype crossing signal.

Note: The only difference between RR_Crossing_Bell_SW_HIGH and RR_Crossing_Bell_SW_LOW is whether the enabling switch is HIGH or LOW. The choice is often dictated by the sensor, mechanism, or switch you use. These are provided to demonstrate the ease of customizing activation as needed.

CROSSING LIGHTS & BELL FREE RUNNING ACTIVE LOW

Sketch folder: RR_Crossing_Bell_SW_LOW

Setup: Sound setup 1 [photo 25]

Control: pin 14 (A0)

Connected pins: led_pin1-11, led_pin2-12, bell_pin-15 (A1), silence_pin-16 (A2), control_pin-14 (A0)

This project alternates the flashing of two LEDs. Flash rate depends on setting **delta**, which defaults to 745.

A simple on/off switch is connected to control_pin- 14 (A0). The project switches ON when the control_pin is LOW.

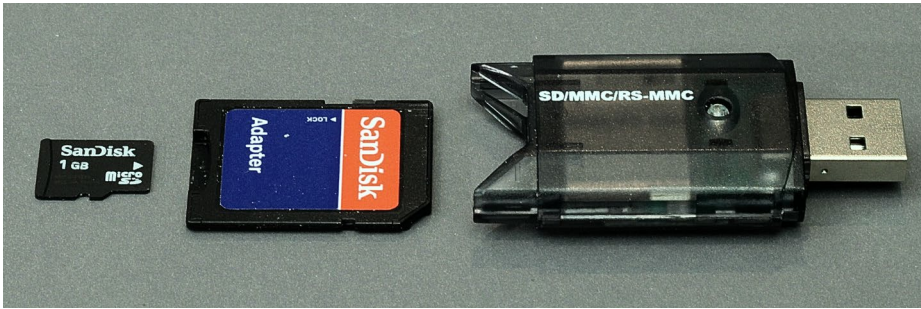
The crossing bell runs at an independent rate, simulating a more modern style of prototype crossing signal.



• INDEX

• TABLE OF CONTENTS





27. Micro SD memory card adapters to USB.

Place the sounds clips on the micro SD memory card “in the root,” which means do not put the sound files into any folders.

First, re-format your micro SD memory card, then load the files in order, one at a time, from the SoundSetup1 folder in the bonus downloads for this article.

What do HIGH and LOW really mean?

In the world of digital electronics, HIGH means 5.00 volts exactly and is interpreted as one (1). LOW means 0.00 volts exactly and is interpreted as zero (0).

In practical terms, however, an Arduino interprets any voltage level above 3.5 Volts (3.5-5.0) as HIGH and any voltage below 1.5 volts (0.0-1.5) as LOW. Ground is always considered LOW.

In all the sketches presented in this article, care was taken to enable an internal Arduino circuit to set all unconnected, open inputs to the board so they default to HIGH. If this action were not taken, a disconnected open input could not reliably be read as either HIGH or LOW. ■



• INDEX

• TABLE OF CONTENTS



PLAY RANDOM SOUND TRACKS

Sketch folder: Random_Sound_Clips

Setup: Sound setup 2 [photo 27]

Control: pin 14 (A0)

Connected pins: (no other pins needed)

This project plays a set of mp3 sound tracks stored on a micro SD Memory Card in random order. A simple on/off switch connected to pin 14 (A0) will enable playing.

Place the sound clips on a micro SD card in a folder labeled mp3. Name them 0001.mp3, 0002.mp3, 0003.mp4, 0004.mp3, etc.

The sketch has two values you can set: **num_clips** (default 9) sets the total number of sound clips to select from on the SD memory card, and **sound_start_delay** (default 0) sets how many minutes to delay before starting to play after turn on.

PLAY SELECTED SOUND TRACKS

Sketch folder: Selected_Sound_Clips

Setup: Sound setup 2 [photo 27]

Control: pin 14 (A0)

Connected pins: select_pins-2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

This project plays one of a set of pre-recorder mp3 sound clips stored on a micro SD Memory Card by pulling a chosen pin LOW to ground. A simple on/off switch connected to pin 14 (A0) enables playing.

Place the sound clips on the micro SD card in a folder labeled mp3 and named 0001.mp3, 0002.mp3, 0003.mp4, 0004.mp3, etc. corresponding to the order of the pins defined by select_pins.

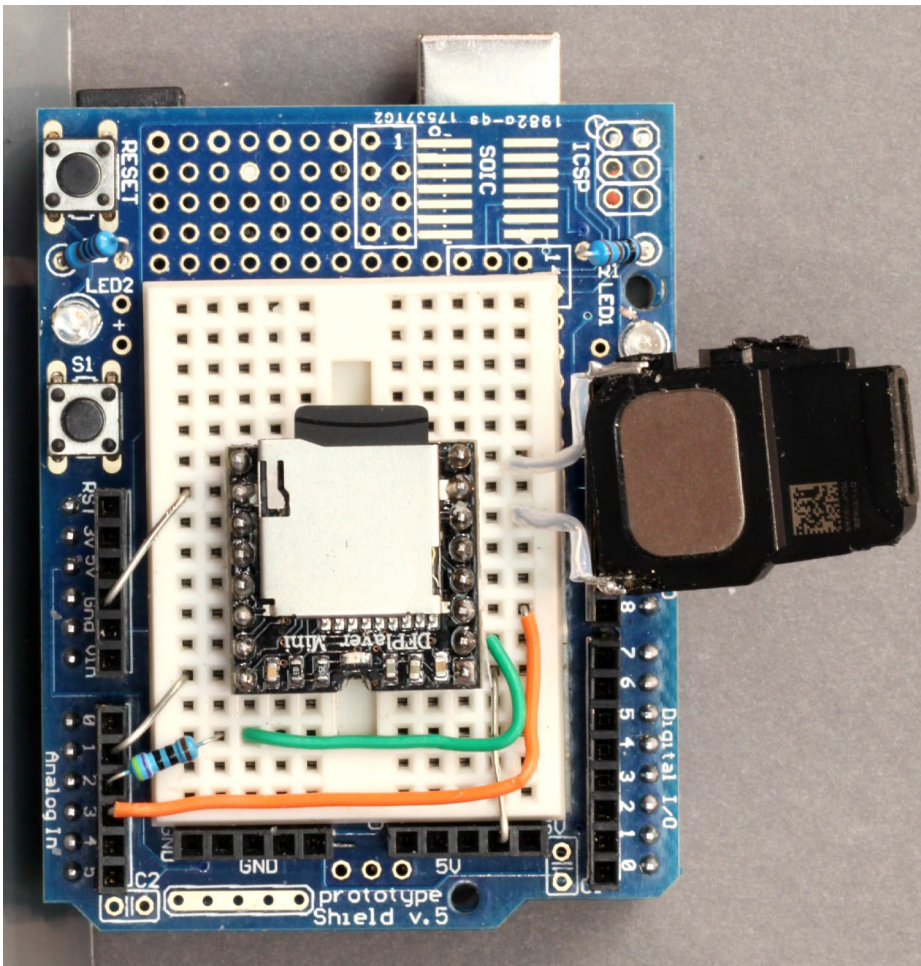
Pulling pin 3 LOW plays clip 0002.mp3. Sketch value **num_clips** (default 9) sets the total number of sound clips/tracks on the SD card, and **sound_start_delay** (default 0) sets the number of minutes to delay before starting to play after turn on.



• INDEX

• TABLE OF CONTENTS





28. Sound setup 2, using an iPhone cell phone speaker. You can get an iPhone speaker for a few dollars on Amazon at this link: [a.co/564x5AP](https://www.amazon.com/dp/B0758Z8Z8Z). Cut off the plastic housing extension and seal the small hole left with some epoxy or putty, leaving the rest of the speaker as shown here. These tiny speakers produce excellent sound.



[Click here for reader comments](#)



• INDEX

• TABLE OF CONTENTS



PLANNED SOUND: CREATE A THUNDERSTORM

Sketch folder: Thunderstorm

Setup: Sound setup 2 [photo 27]

Control: pin 14 (A0)

Connected pins: lightning_pin - 3, lightning_pin2 - 4

This project plays a set of mp3 sound tracks stored on a micro SD Memory Card to simulate a thunderstorm approaching and then passing.

A simple on/off switch connected to pin 14 (A0) enables playing.

You can set up a start delay by changing `lightning_delay` in the sketch to the number of minutes you want to pass after the initial turn on.

You can also set `next_storm_delay` to the number of minutes before the next storm arrives!

Note: While you can use this as a sound-only sketch, it does include synchronized switching for firing two independent slave strobe lights. These are activated with a relay connected to pin 3 and pin 4.

The entire sketch with sound and lights simulates a thunderstorm approaching and passing by. Two strobes are needed to account for the long re-charging time of the strobes. You will notice closely timed lightning bolts in your area!

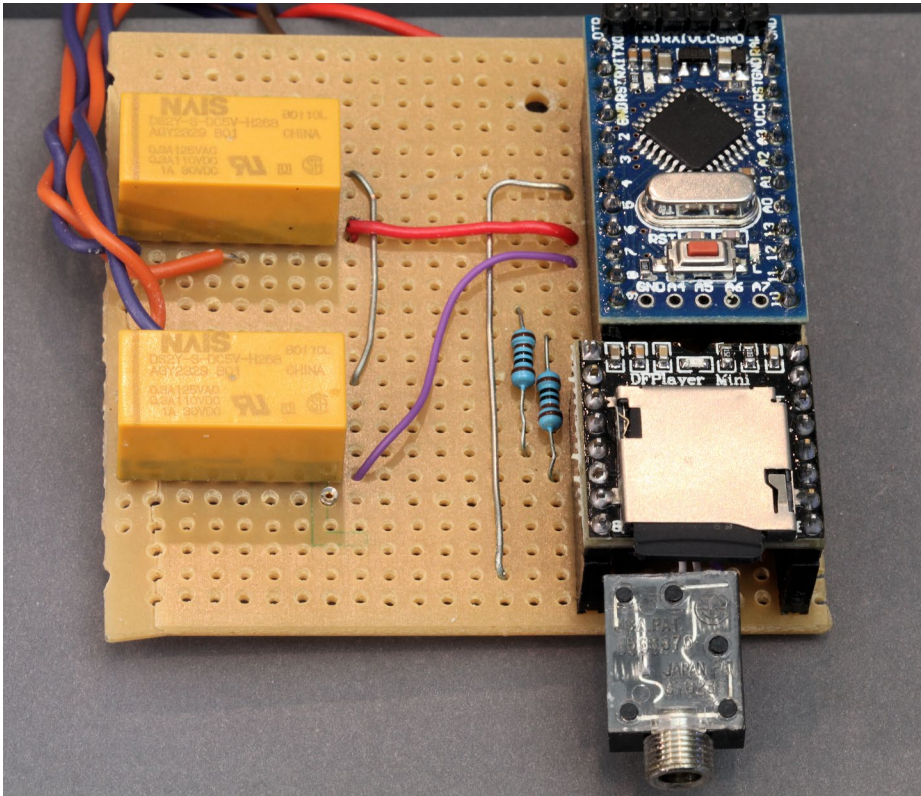
Load all the sound files onto your micro SD Memory card and try to stay out of the rain! When I have given clinics on modeling with sound, I often will stage things so the thunderstorm will arrive and pass somewhat unannounced in the middle of the clinic!



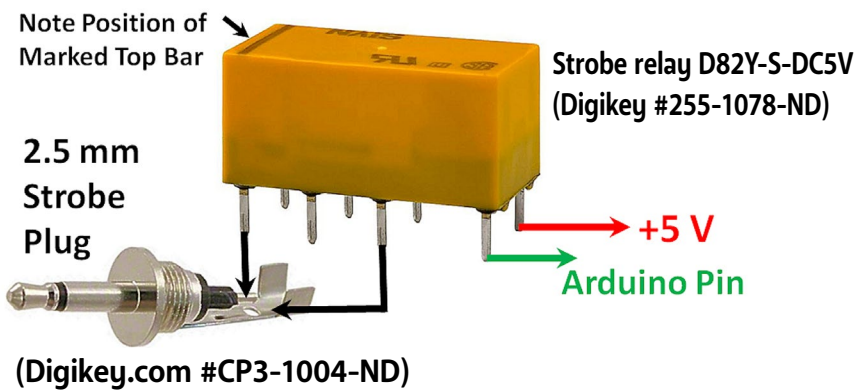
• INDEX

• TABLE OF CONTENTS





29↑ and 30↓. Thunderstorm components all wired and ready to create a storm above. Relay wiring diagram shown below.



• INDEX

• TABLE OF CONTENTS

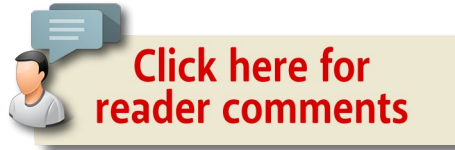


31. Self-contained strobe for lightning/thunderstorm.

Amazon link:
a.co/ba5G273



Be sure to check the [subscriber bonus downloads](#) for this month's issue. There's a lot of supplemental material in there to go along with this article. Have fun! ☑



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. ■



• INDEX

• [TABLE OF CONTENTS](#)

