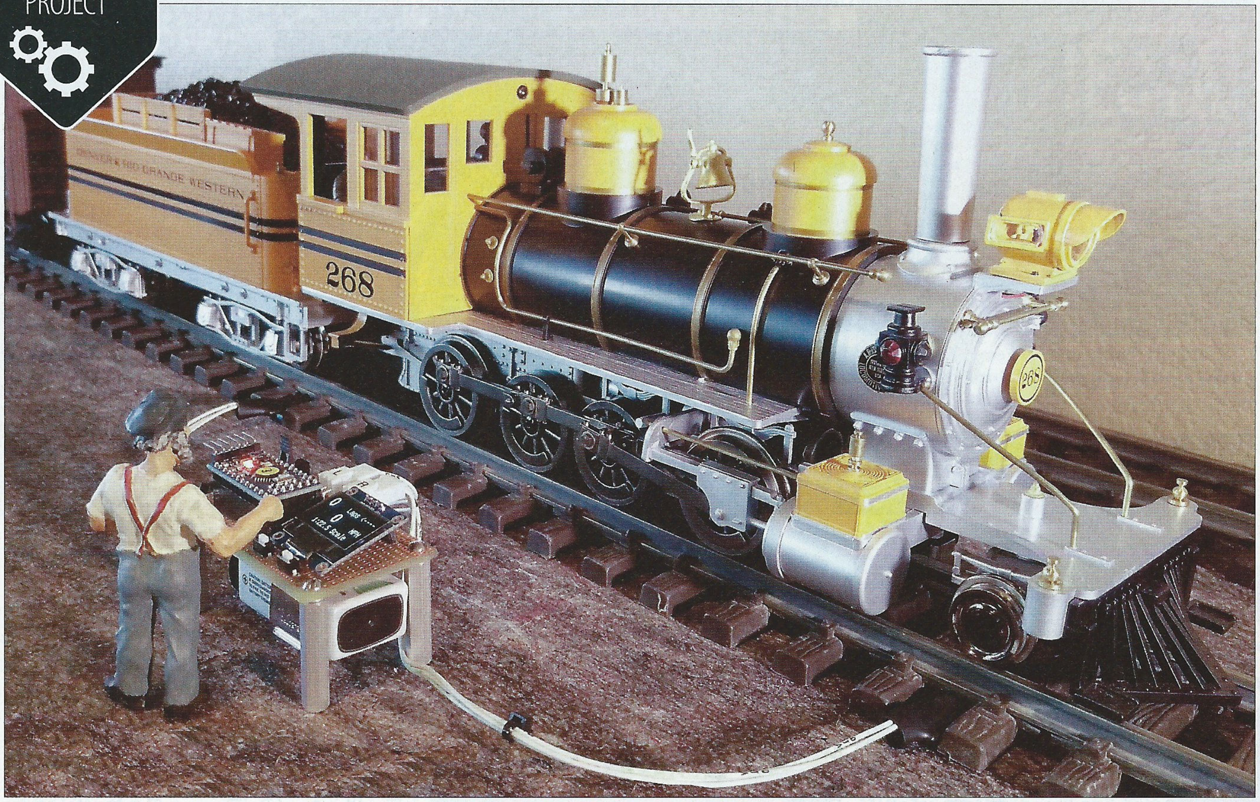
INTERMEDIATE PROJECT



A technician checks the locomotive's speed as it passes over sensors in the track. Build this simple speedometer for your own railway.

SPEED TRAP

Build a speedometer for your railway using an Arduino

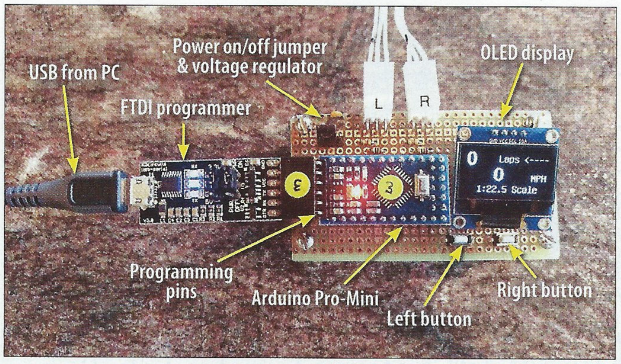
by Del Tapparo | Fort Collins, Colorado | PHOTOS AND ILLUSTRATION BY THE AUTHOR

"How fast will your train go?" You probably hear that question from every kid that sees your railroad. You may have asked yourself, "How fast should I be running my trains for prototypical speeds?" I usually set the speed of my model trains to something that is pleasing to the eye and makes sense for the type of locomotive I am running and the surrounding scenery, and that is good enough. But I'm always looking for ways to apply my electronics hobby to my garden-railroading hobby. My recent interest in Arduino microcontrollers had me looking for projects to learn and experiment with, so I decided it would be fun to build a device to measure the scale speed of large-scale trains.

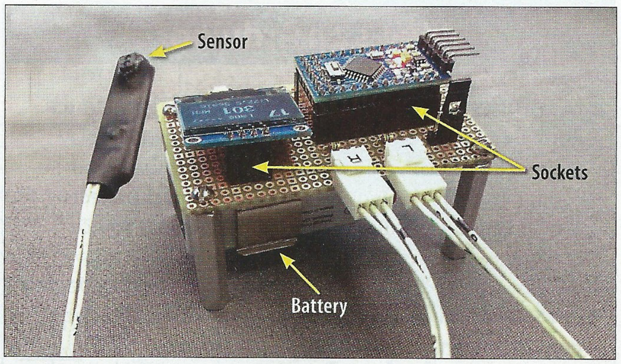
The basic idea is pretty straightforward. A trackside "speed trap" will time the train between two points spaced a known distance apart. Speed = distance ÷ time, so we need to measure the time, then calculate the actual speed over the known distance. Scale speed can then be calculated based on the desired model scale (1:20.3, 1:29, etc.). A microcontroller and a couple of light sensors will do the job.

Hardware and construction

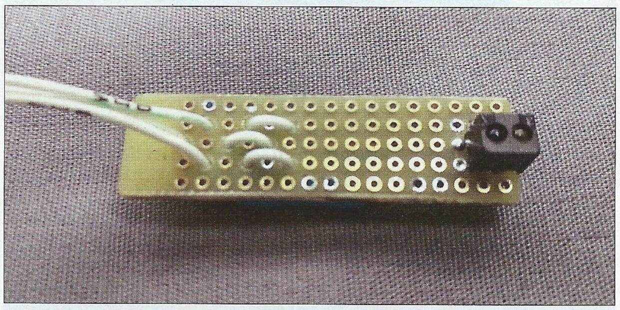
I wanted to use two light sensors—one to start the timing and the other to stop it. These sensors would be mounted in the track between the ties. As the train passes over a sensor, the light level on the sensor will change. However, since our large-scale trains sit about one inch above the ties, that leaves a lot of open space for light to get in and confuse the sensor. After an extensive search, I decided to use a reflective photo sensor. This device shines an infrared (IR) light source straight up, and has a photo sensor right next to it to measure any IR light that is reflected back from a



1. Scale speed and lap count are displayed on the small OLED display. The right button is used to change scale or direction of operation; the left button flips the display. The Arduino Pro-Mini version requires an external FTDI programmer. However, most other Arduinos have this on-board and use the USB directly.



2. Construction was done on a proto board, with wiring connections on the bottom. Standoffs provide a stable platform and space to hide the battery.



3. Each sensor is mounted on a piece of proto board.

surface (like a passing train). Thus, the predominate light level being measured would be from the IR source itself, not the surrounding visible light.

I found the RPR220 Reflective Photo Sensor and began to experiment. Even though it uses IR, and claims to filter visible light, it is still affected by changing ambient light conditions in the room or outdoors, but it works nicely. An object passing over it may cause an increase or a decrease in the IR light level detected, depending on current lighting conditions. The key is to look for a change in the light level to trigger the timing.

Tiny OLED displays are now available that can display a lot of information in about one square inch of space. They communicate with the Arduino over the I2C bus, which consists of only four wires: power, ground, clock, and data. Lucky for us, we don't need to know anything about I2C. We just download a free I2C software library that someone else has already developed for us, and use that. Software libraries are a strong point in using the Arduino system. With a few simple commands, we can display text anywhere on the 128 x 64 display, change fonts, and change font sizes.

As you know, there are many different scales of trains running on our gauge-1 track, so I added a push button to step through a menu of scales (1:20.3, 1:22.5, 1:24, 1:29, and 1:32). Then, after realizing the display could be viewed from either side of the track, I add another push button to flip the display, if needed.

Power for the project is via a 9V battery and a simple on/off switch. The 9V is regulated down to a Vcc of either 5.0V or 3.3V (depending on your Arduino) by an external voltage regulator. Although the Arduino has a built-in regulator that can be used via the RAW input, it doesn't have enough current capability to power the light sensors. The whole thing is built on proto board (a circuit board with a grid of plated holes, allowing you to interconnect components with wiring—**photo 1**). The Arduino and the OLED are mounted in sockets (**photo 2**) to allow easy replacement (or reuse). The battery is held in a clip mounted to the underside of the board with double-sided foam tape.

The sensor modules contain the photo sensor and a three-wire cable that connects to the main board via locking connectors (**photos 3** and **4**). I shrink-wrapped the entire module, then cut out a window for the sensor and sealed the end with silicone adhesive to give it some protection from

Speed trap parts list

Main board assembly (1 each)

- Arduino, Pro-Mini (*www.banggood.com*)
- OLED Display, 128 x 64 (*www.banggood.com*)
- LM78L33 or LM78L05 voltage regulator
- Tactile switch (momentary push button for menu)
- 47K ohm, 1/8W resistor
- 75 ohm, 1/8W resistor
- 0.1 microfarad capacitor
- 3-pin connector (male)
- 3 pin header & jumper (or SPST on/off switch)
- Battery, 9V rechargeable NiMh
- Battery clip w/wires
- Battery holder
- Header sockets to mount Arduino and OLED Display
- Proto board, 1.7" x 3"
- Wire, 26 ga. solid
- 4 PCB standoffs and screws (1¼"L, 4-40 thread)

Sensor modules (2 each)

- RPR220 reflective photo sensor (*www.eBay.com*)
- Proto Board, 1⅞" x ½"
- Wire, 22 ga. stranded, 3 ea, 10"L
- 3-pin connector (female)
- Heat-shrink tubing (¾" dia.)

Programming

- FTDI programmer (for Arduino Pro-Mini only—not required for Arduino Uno)
- Arduino IDE software (*www.arduino.cc*)
- U8glib.h Library (Software for OLED Display, *https://github.com*)

What's an Arduino?

"ARDUINO" IS AN ITALIAN micro-controller board capable of interfacing with many kinds of sensors and display devices. It is a relatively easy way to get into embedded computer programming and simple electronics projects. Arduino has been around since 2003 as an open-source community. As a result, the hardware and software have become standards for hobbyists and makers worldwide. You can buy not only genuine Arduino boards, but reproductions from China at low prices.

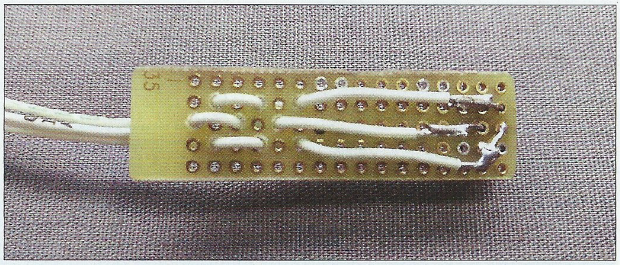
Arduino programming uses a simplified version of the "C" programming language. "C" makes use of free software libraries, which means that for nearly any sensor, hardware accessory, or function you can think of, somebody will have already written a library for it that you can download and use free, making it simple to get your project started.

What do you need to get started? Arduino boards come in many sizes and configurations. The best way to start is with a genuine Arduino Uno R3 board at a cost of about $25 from US vendors, or you can get a Chinese knock-off for under $5 with free shipping and slow delivery (4-5 weeks). While the Chinese products are low cost, they have no documentation, no instructions, and no support. There is an abundance of information available online but buying from a US distributor is advised for the first-time user.

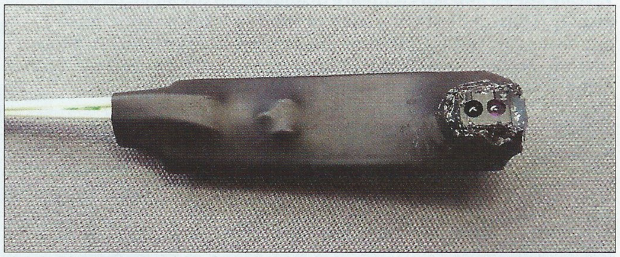
Genuine Arduino distributors include *www.Adafruit.com, www.Sparkfun.com, www.SolarBotics.com, www.Digi-key.com, www.mouser.com,* and more. Chinese reproductions can be had from *www.Banggood.com* and *www.eBay.com*

Arduino programs are called "sketches" and they are composed, edited, and uploaded from your PC to your Arduino via USB, using an IDE (Integrated Development Environment). The IDE is available as a free download from *www.arduino.cc,* where you will also find complete reference material and examples.

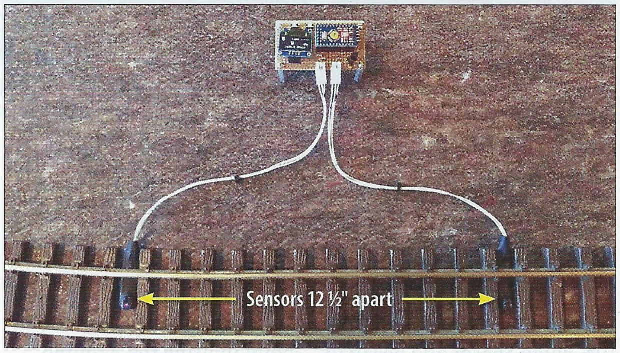
Type "Learn Arduino" into your browser and you will find everything you need. Search for "Arduino" on YouTube and you will find hundreds of tutorial videos, application examples, and reviews.



4. Solder connections for the sensors are made on the bottom side of the proto board. Only three wires are needed for each. Wires are woven through the board to relieve any stress on the soldered joints.



5. To help protect them from dirt, each sensor and its little board are wrapped in heat-shrink tubing.



6. The speed trap module is set up trackside with sensors between the ties, placed exactly 12.5" apart, which is well suited for LGB tie spacing (12 ties).

dirt and track ballast (**photo 5**). The sensors should be placed exactly 12.5" apart in the track (12 LGB ties apart), with the sensors centered between the rails (**photo 6**). This distance value can be changed in the software if necessary. As built, this project is intended for temporary use outdoors or permanent use indoors. Better packaging of the main board would be required for long-term outdoor use.

Operation

On power-up, the display will show speed for a 1:22.5-scale train traveling from left to right as indicated by an arrow. A passing train will display the scale speed in the center of the display, and update a lap counter in the upper left-hand corner of the display. As the train passes over the light sensors, the tip of the locomotive trips the first one to start the timer, then the second to record the elapsed time. The sensors are then deactivated for five seconds while waiting for the remainder of the train to pass by.

To change the direction of train travel, hold down the right button under the display for one second. To change scales, just give momentary button presses until you see your desired scale. A momentary press of the left button will flip the display upside down for viewing from the opposite side of the track.

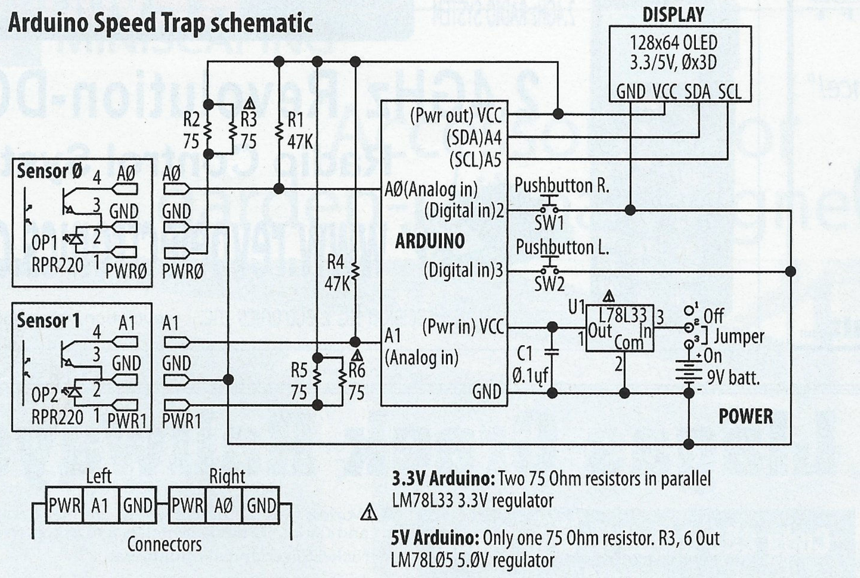
Software

The program runs in a continuous loop, executing routines as follows:

- Check ambient light (updates ambient-light levels every 10 seconds)
- Change scale? (check for a right-button press)
- Flip screen? (check for a left-button press)
- Read light sensors (looking for a level change in the first sensor to start timing)

When the first sensor is tripped, the elapsed time to tripping the second sensor is recorded. Scale speed is calculated, the lap counter is incremented, and the display is updated. We then wait for the rest of the train to pass, which

Arduino Speed Trap schematic



would otherwise cause much confusion with the all of the multiple trips from train cars. End of train is determined by the second sensor remaining at its ambient light level for at least five seconds, then the speed trap is ready for the next train. Sensor-trip thresholds are determined by comparing the previously stored ambient light condition with the current sensor reading.

This Arduino sketch (Arduino's name for a program) requires downloading the U8glib.h OLED display library from *https://github.com*

Online extras

The source code that will be used by the Arduino IDE can be downloaded from our website at *www.GardenRailways.com*. Type "Arduino" into the search box.

Conclusion

While a "speed trap" is not a must for your railroad, this is a great project for learning about microcontrollers, programming, and electronics construction. You can apply these principles to other projects. If nothing else, you can demonstrate to your railroad buddies that you run your Shay at a prototypical scale seven miles per hour ! (Just tell the kids, "Sorry, that's as fast as it goes.")