Technology Focus by Richard Cadena

This month: DMX Shield for Arduino

> "... it only has a few lines of code, so it's easy to understand and it's a great introduction to programming. It's also a good tool for lighting techs to learn about LEDs and pulse-width modulation (PWM)."

A couple of months ago, I spent a morning becoming familiar with a tiny computer, technically a microcontroller, called Arduino (see *Technology Focus: Love, Tech Style, LSi January 2014*: http://bit.ly/love-tech) I've since had a chance to dive deeper and also spend a little time with another small computer, or microcomputer, call Raspberry Pi. What I've found is that there are lots of little lessons to be learned from these mini-computers that directly apply to entertainment technology.

If you want a fun way to learn more about your craft, I suggest you start with the *Make: Getting Started with Arduino kit* (**http://bit.ly/Make-GSA**). It has everything you need to get up and running with the Arduino Uno. (NB. The Uno is one of the entry-level versions of Arduino.) This microcontroller is based on the Atmel ATmega328 chip with "reduced instruction set computing" architecture - a so-called RISC computer. That's what makes it smaller and less power-hungry than your desktop or laptop computer. It's similar to the type of processor found in your mobile phone or tablet.

Besides some cables, a battery case, solderless breadboard, jumper wires, mini switch, some LEDs and resistors, the kit comes with a 118-page book entitled *Getting Started with Arduino*. It's an easy read, and it walks you through the whole experience with a lot of hand-holding. Also, the online forums (**http://forum.arduino.cc**) are very active and people are quick to respond to your questions and to help you resolve problems.

To get started you have to download the IDE software (for "integrated developer environment") to your computer, which then becomes the interface for programming the Arduino. The book walks you through the process and once it's installed, there are some very simple programs to help you get acquainted, the first of which is turning on an LED.

This program, or "sketch" in Arduino parlance, is called Blinking LED, and it only has a few lines of code, so it's easy to understand and it's a great introduction to programming. It's also a good tool for lighting techs to learn about LEDs and pulse-width modulation (PWM).

PWM is a method of dimming LEDs, and if you're unfamiliar with how it works, the book explains it very well. The simple version is that an LED is turned on and off repeatedly at a very high rate of speed, or frequency. The percentage of time that it's on compared to the time it's off, which is called the "duty cycle," determines the width of the pulse and the perceived brightness. If the modulation frequency, or speed of the switching, is high enough, then the human eye doesn't see it turning on and off; it appears to be constantly lit even though it's actually blinking. If the modulation frequency is too low, then the flicker is obvious.

You can find the frequency at which you can no longer perceive any flicker by changing a couple of numbers in the Blinking LED program. There are two lines with the command "delay;" one which controls the 'on' state and another which controls the 'off' state of the LED. The default values are 1000 milliseconds (or one second) on and 1000 milliseconds off, which is a duty cycle of 50% with a modulation frequency of 0.5 cycles per second or 0.5 hertz (Hz). (That's one cycle per two seconds or 0.5 cycles per second.) At that very low frequency, the LED looks more like a beacon than a dimmed LED.

To change the frequency, we can change the values of delay in the code. If we use 100ms (0.1 seconds) for both the on and the off values, that changes the frequency of modulation to 5Hz. It's still very obvious at that frequency that the LED is blinking. But if we change both values to 10ms, which is a modulation frequency of 50Hz, you can no longer see the flicker and the LED appears to be 50% dimmed. After some experimenting, I figured out that the threshold for seeing flicker was between 50Hz (no flicker) and 45Hz (flicker), which corresponds to values of 10ms and 11ms in the code, respectively.

But if you record video of the LED at that frequency, you will find that you can see flicker in the playback that you can't see with the naked eye. I use my iPhone for this purpose because it's handy, and I believe that if I can get good results with an inexpensive sensor, like that in an iPhone, then a good camera with a better sensor will work even better. What I found by experimenting with the PWM frequency is that the best results are when the frequency is a whole number multiple of the frame rate of the camera, or the frequency is much higher than the frame rate of the camera.

My iPhone has a frame rate of 29.97 fps, and by changing the values of the on and off times to 16.7ms, the LED flicker will approximate the frame rate of the camera. Unfortunately, the value of delay using the Uno can only be whole number, so instead of 16.7ms, I used 17ms, which yields a frequency of 29.4Hz. This frequency is easily seen as flashing by the naked eye, but by changing the times to 1 or 2ms, which corresponds to 500 or 250 fps respectively, the video is very smooth and free from artifacts.

You can get much higher modulation frequency by changing the command from delay to delayMicroseconds.

That changes the delay units of measure from milliseconds (1/1000th of a second) to microseconds (1/1,000,000th of a second).

For the best results, get the camera very close to the LED emitter, and then you will see how it interacts with the camera: you'll be surprised at what you see. Sometimes you can see outright flicker, and other times you see more subtle, slow rolls and other artifacts.

It's also interesting and informative to program a 10-second fade to black to see how the resolution affects the smoothness or steppiness of the dimming curve. You can experiment to find out how much resolution it takes to make it dim smoothly. There's an example program that you can download from the Arduino website (**http://plasa.me/108fw**) but it uses the analogue outputs on the board. In real life, we use DMX for this purpose.

You can use Arduino to transmit and receive DMX, but you need a daughterboard, or in Arduino parlance, a "shield". This is a small PCB with circuit parts and connectors that attaches to the main Arduino PC board.

There is an entire area of the Arduino website called the "playground," and part of it is dedicated to DMX, RDM, and ArtNet (**http://plasa.me/gh58n**). It's full of good information that's easy to understand and follow. A great place to start learning more about DMX is a playground page titled 'The DMX Protocol' (**http://plasa.me/qhvq8**). It covers the bit timing, slot timing, and packet timing.

Another web page in the playground covers the basics of DMX shields for Arduino (**http://plasa.me/f11i4**). It describes the chips that are used to transmit and receive DMX data (MAX485 transceiver chip), the PC board layout, the connector pinout, and it provides a "DMX Library for Arduino" with links to example sketches. These pages are great if you want to roll your own DMX shield, or if you prefer to buy them pre-assembled, there are links for that too.

Another good source of information about Arduino DMX shields is Matthias Hertel's website (**www.mathertel.de/Arduino**). I followed his instructions and bought some Maxim MAX481CPA chips for U$2.91 each, some Everlight 6N137 optocouplers for U$1.04 each, and some Murata isolation transformers for U$8.12 each, all from Mouser Electronics (**www.mouser.com**), to build a shield. I started to research where I could have some PC boards etched but I decided instead to buy pre-made PC boards directly from Matthias (he "sometimes" offers both the boards and the finished shields, though he cautions that he doesn't always have them available). In this case, I paid him through PayPal and I received a finished shield and enough parts to build a second shield in about a week and a half.

Once I had the DMX shield, I downloaded the project files, which included a DMX serial library file and three example sketches. The sketches require you to connect a wireless breadboard or a PC board that has one red, one green, and one blue LED, each of which is powered from Arduino through a 250 ohm resistor. One of the sketches, called DMXSerialFlow, fades between several different combinations of colours by cycling each LED through a fade up and fade down. Another sketch, called DMXSerialSend, essentially does the same thing except it cycles through six distinct colour combinations. The third sketch, DMXSerialRecv, turns the Arduino into a DMX receiver that can interpret the DMX signal sent by the first two sketches.

Raspberry Pi is a more powerful but still small computer, and I've barely scratched the surface of its capabilities. Both of these platforms are designed to bring out the artist, designer, engineer and tech in you. They are gateways to all of the hardware and software we use in our industry every day. They can be used for control, media servers, robotics and so much more. It's a great big technology world out there, but it can be conquered by the smallest of technology.