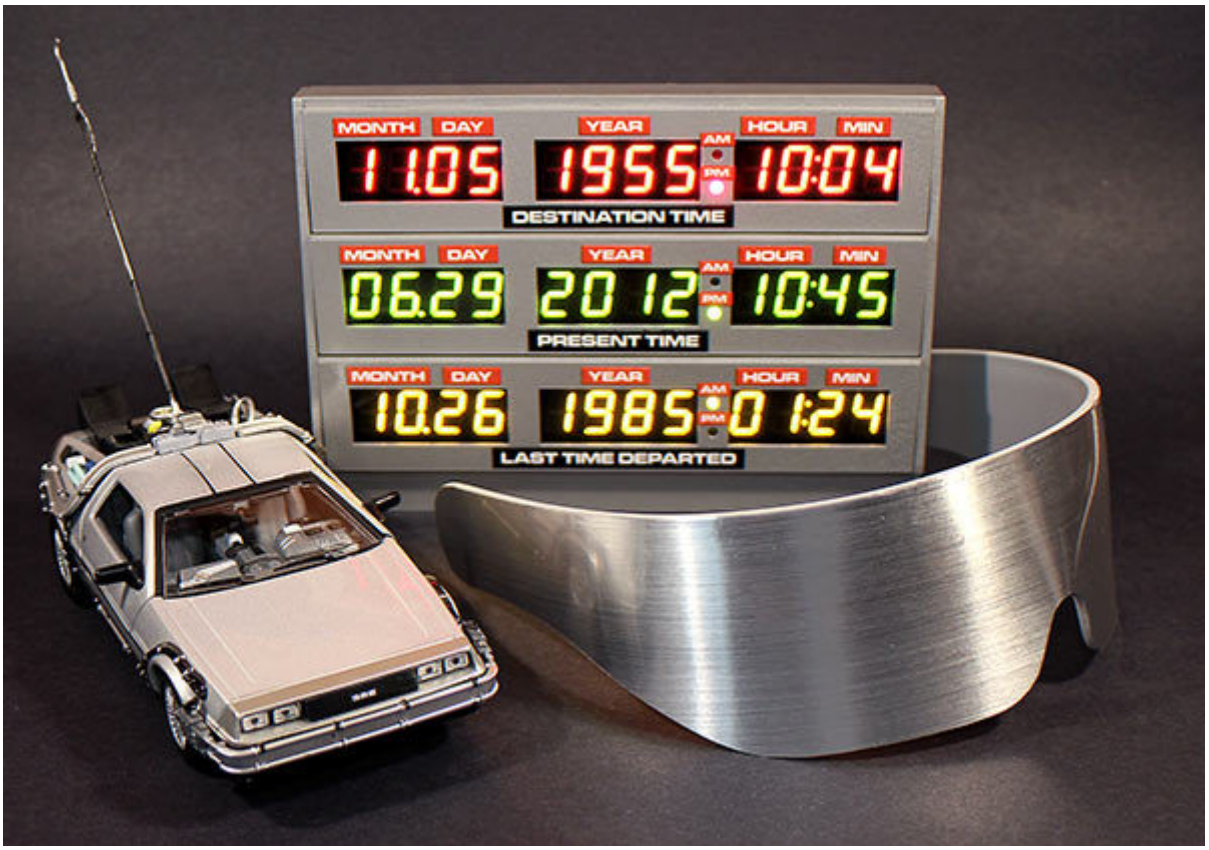
DeLorean Time Circuit

Created by Phillip Burgess



https://learn.adafruit.com/delorean-time-circuit

Last updated on 2025-01-22 12:13:29 PM EST

Table of Contents

Background 3

Design Liberties 4

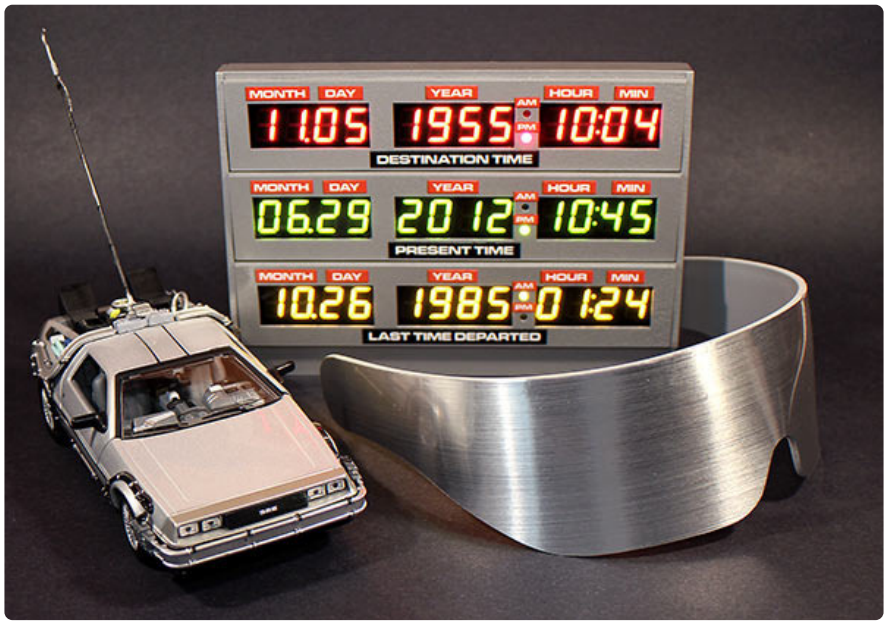
Circuit Trickery 5

Fabrication 6

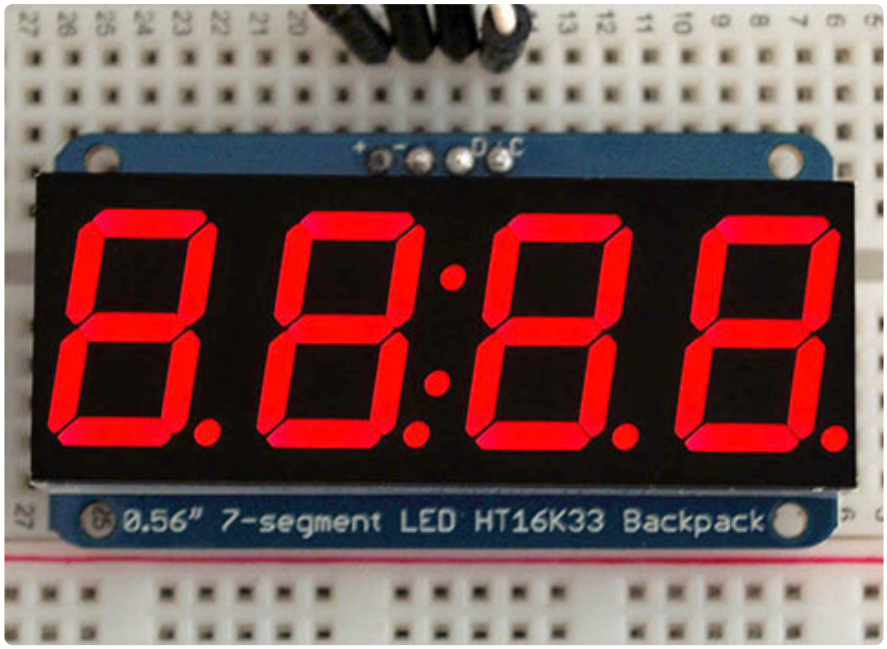
Wrap-Up and Resources 9

- Resources
- Code

Background



Getting the big question right out of the way: no, we won't be selling these. Legally, we can't, for trademark reasons. But it's generally okay for anyone to create replica props for their own personal use, so we hope this writeup will inspire some cool projects among our customers...



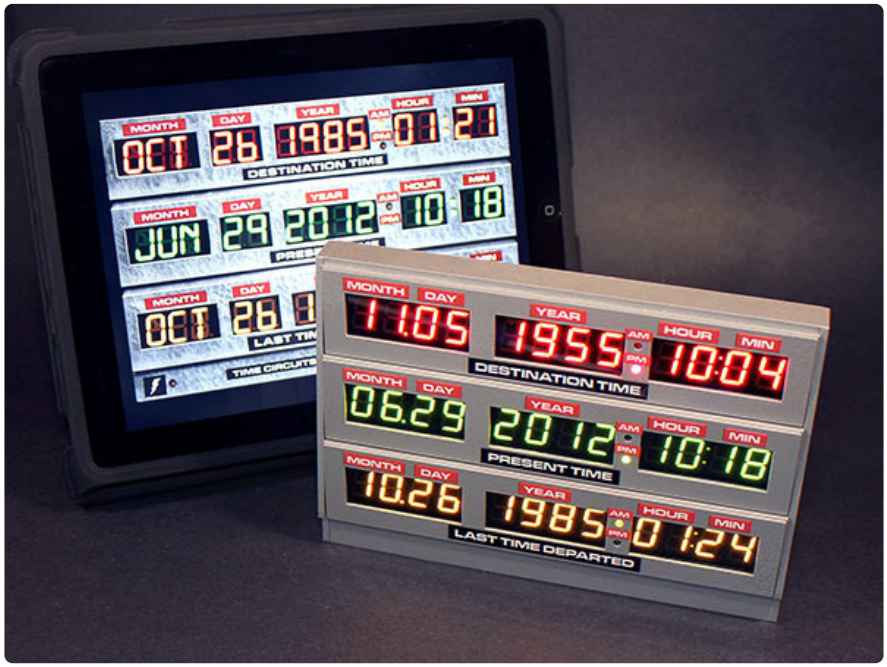
From the moment these (http://adafru.it/878) LED (http://adafru.it/880) displays (http://adafru.it/879) made an appearance on our weekly Ask an Engineer (https://adafru.it/

aH0) show, comparisons were being made to the DeLorean time circuit from the Back to the Future films. It was a moral imperative then to make a demo! If you're handy with Arduino and some shop tools, you should be able to pull off something similar (better, even), or adapt the ideas to other projects. This was quickly built in fun, so please don't expect the same level of polish as a finished product tutorial.

Design Liberties

When accepting this assignment, I might've failed to mention a small detail to the boss folks: I don't own a car, let alone a DeLorean, for displaying the finished prop. Instead, mostly inspired by Jeri Ellsworth's NES purse (https://adafru.it/aH1), I had this goofball idea of a slim, battery-powered device that could be installed and photographed in ironic settings: on a bicycle, on public transit, hung from a Flavor Flav (https://adafru.it/aH2) necklace as "bling," and so forth.

While the general idea could have been accomplished quickly and easily with an iPad running the Flux Capacitor™ (https://adafru.it/aH3) app, I wanted to preserve somewhat the staggered design of the original, and it had to have real 7-segment LED displays...there's no substitute for seeing the genuine thing. In much the way that nixie tubes (https://adafru.it/aH4) have a certain vintage coolness about them, LED displays too are reaching a nostalgic threshold, iconic of 1980s technology.



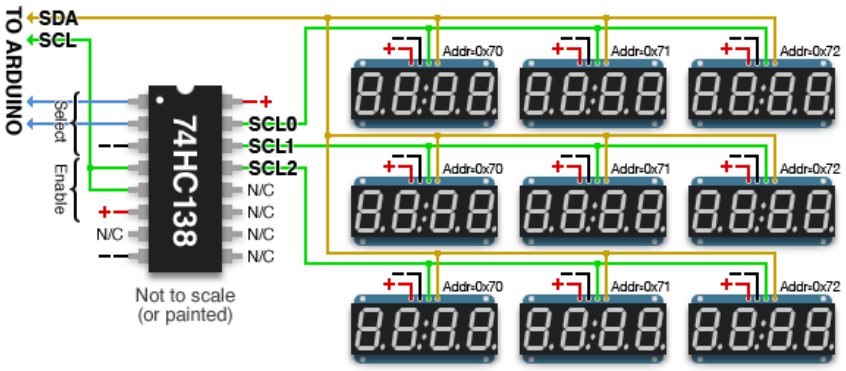
Using stock parts required some design compromises. The date and time formats would be changed to fit these 4-digit displays (the film prop used back-painted glass fakes for the month display, with some segment changes being physically impossible, making a 100% match unattainable anyway...iPad wins there). Also took liberties with some LED colors and various spacings, but overall the piece is still highly recognizable.

For the sake of a quick demo, I had to cut this short. Though all the displays are addressable, the destination and last-departed dates are simply fixed values from the first film; there's no interaction. I may revisit this to add a keypad later, but for now it's all just a fancy clock (it does show the current time accurately, using a ChronoDot RTC (http://adafru.it/255)). Also, the vector files are not available, because they're utter garbage! Creating something of finished kit quality requires many iterations and refinements (https://adafru.it/b33568)...but with a rushed, one-shot piece like this, course corrections would come in the form of a Dremel tool and epoxy putty. If you plan to build one, give it some time and prepare your blueprint carefully.

Circuit Trickery

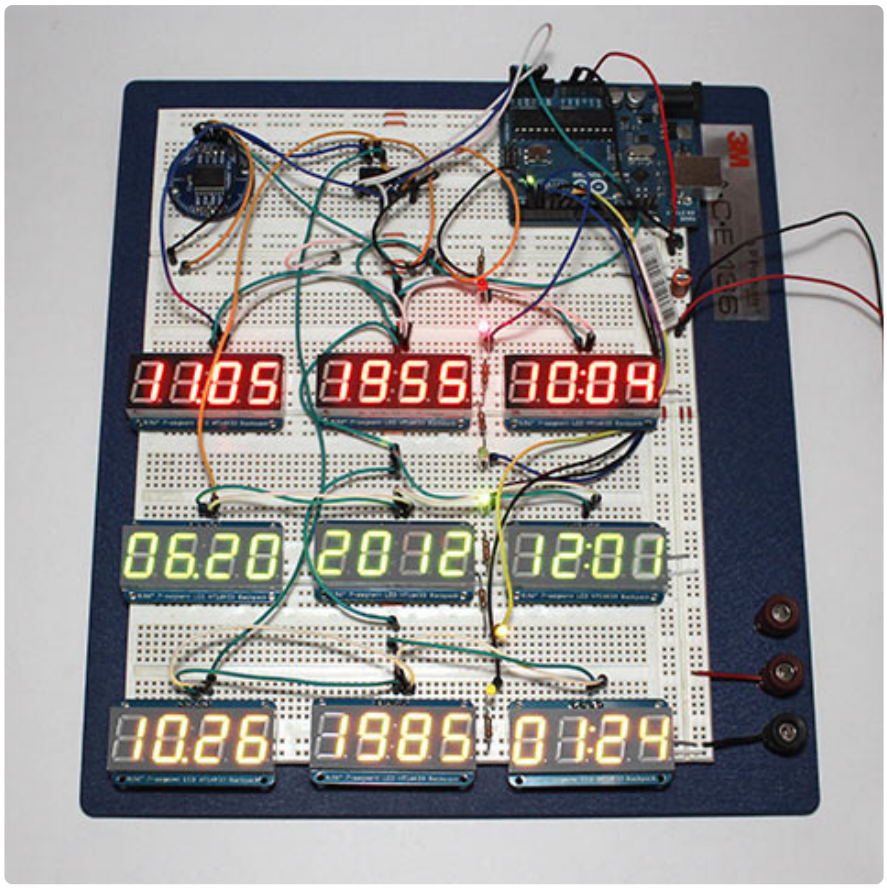
These 4-digit displays can be assigned one of eight fixed I2C addresses via solder jumpers on the back. But the time circuit needs nine displays. A few possibilities were considered, including driving the one extra display "manually" with shift registers, or use a software I2C library and split the displays among multiple I2C buses. Either would require lots of library code changes and some intense concentration, but I was hit with a massive sinus headache at the time and really didn't want to think about it.

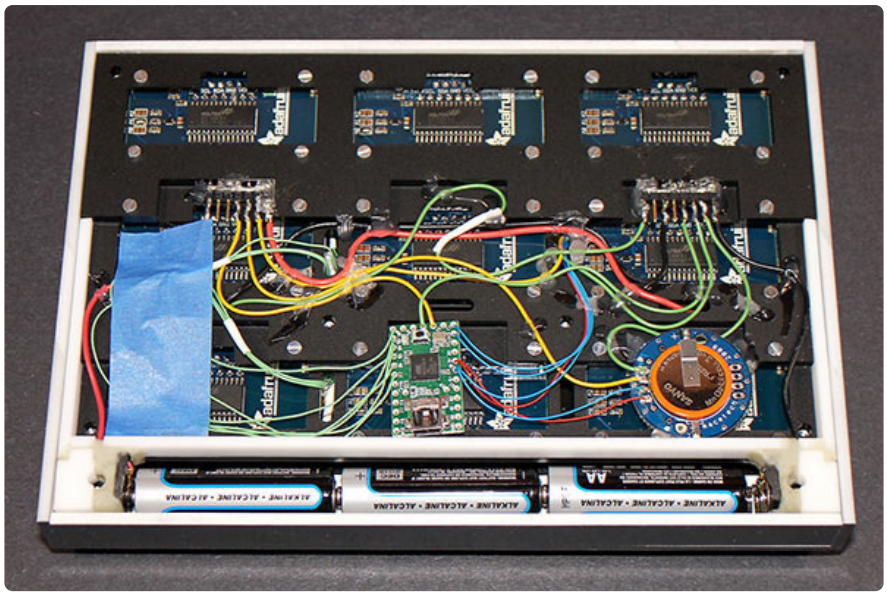
Instead, exploiting the fact that we need just one way, write-only access to use the displays, I used a simple hardware hack to split the I2C bus to communicate with one row of three displays at a time (and saving some code by repeating the same addresses in each row). The I2C data line fans out to all the displays as normal, but the clock feeds the enable lines of a 74HC138 3-to-8 line decoder, and the microcontroller can then select which output line forwards the clock signal. The data on the other I2C buses is ignored without the corresponding clock.



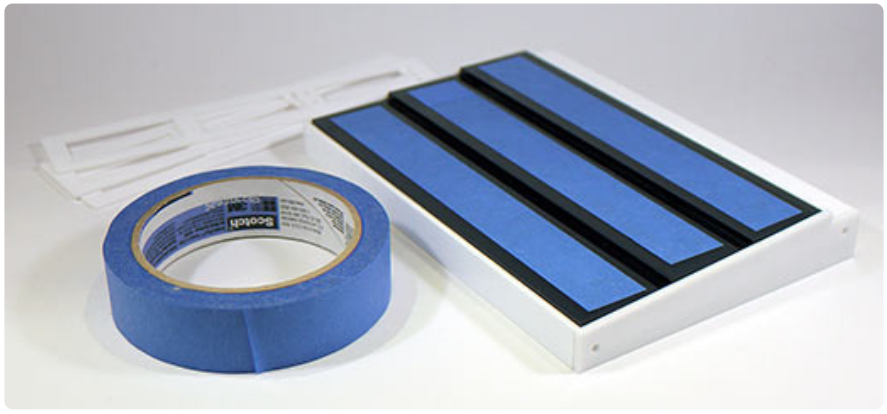
Fabrication

To keep this ultra slim, a Teensy (http://adafru.it/199) microcontroller board was used — a standard Arduino wouldn't fit, not even the headerless Leonardo (http://adafru.it/883). After prototyping the full circuit on a breadboard, all the parts were soldered point-to-point and "dead bug" style inside the case. Power is provided by three AA cells in series — a bit under the ideal 5 Volts, but still sufficient to run everything. The cells fit in the "chin" below the three dates. I'd mail-ordered a special battery holder for this, and then in my rush to complete the project I went ahead and made all the case parts based on the holder dimensions on a web site. Naturally then, with the case already cut and glued, the part that arrived was slightly larger than the dimensions posted. The fix was to break off the battery contacts from the ends of the holder and epoxy putty them directly into the case. This eliminated just enough girth for everything to fit. The remaining electronics were delicately folded into the case with copious amounts of hot-melt glue, tape and swearing.





The case was fabricated from laser-cut acrylic and sprayed with faux hammered metal paint. A metal enclosure would have been more authentic (and more work), but a corollary to "Maslow's hammer (https://adafru.it/aH5)" dictates that when you have a laser cutter, every project appears ideally suited to acrylic.

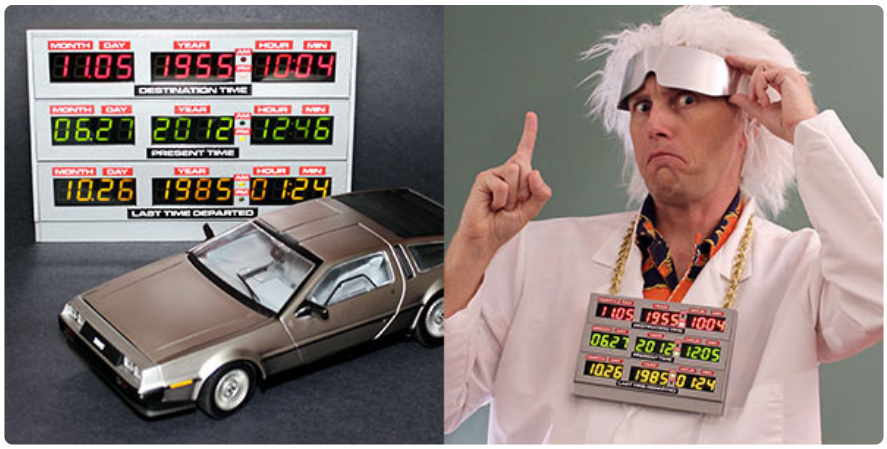


The labels were inkjet printed and made into stickers with a Xyron applicator, trimmed with an X-Acto knife, then painstakingly touched up with a Sharpie marker to hide the white edges. After the labels were applied, the bezels received a thick spray of acrylic sealer, then attached to the front of the case with epoxy.

A classic Dymo labeler (the plastic punched letter kind) might suffice here. In the film trilogy, most of the instruments (including the Flux Capacitor) were labeled that way. But the Time Circuit, being a close-up “hero prop” that required maximum legibility for the audience, had cleanly-printed labels. Sticklers for accuracy might want to take the extra step.

Wrap-Up and Resources

Go for it! If you don’t own a DeLorean, this will still impress your co-workers and look great on your desk. Or maybe you can devise a scheme around Halloween or a cosplay geek-fest like Dragon*Con (https://adafru.it/aH6). Bolt it just below the arc reactor on your Iron Man suit (you do have an Iron Man suit, right?). Or if you have a young son in a stroller, attach the time circuit to the tray, dress junior in mirrored shades and a “life preserver” down vest, while dad dons a Doc Brown getup...you’ll take home all the candy in the neighborhood!



Even if you don’t build this exact item, if it inspires any nifty electronics projects (*cough*Proton Pack*cough*), please share them in the forums (https://adafru.it/forums), bring them to the Saturday night show-and-tell (https://adafru.it/aH7) or

document your build on a site like Instructables (https://adafru.it/aH8). Customer projects are frequently showcased on the Adafruit blog!

Resources

Parts from the Adafruit store include:

- Teensy (ATmega32u4 USB dev board) (http://adafru.it/199)
- ChronoDot Ultra-precise Real Time Clock (http://adafru.it/255)
- 0.56″ 4-Digit 7-Segment Display w/I2C Backpack – Red (http://adafru.it/878)
- 0.56″ 4-Digit 7-Segment Display w/I2C Backpack – Green (http://adafru.it/880)
- 0.56″ 4-Digit 7-Segment Display w/I2C Backpack – Yellow (http://adafru.it/879)
- Diffused Red 3mm LED (http://adafru.it/777) (note: film prop used yellow LEDs on destination time)
- Diffused Green 3mm LED (http://adafru.it/779)

Additional parts acquired from Digi-Key include:

- Diffused Yellow 3mm LED (https://adafru.it/aHa)
- SN74HC138N 3-to-8 line decoder/demultiplexer IC (https://adafru.it/aHb)

Elsewhere:

- Here's an excellent DIY flux capacitor tutorial (https://adafru.it/aHc).
- The Replica Prop Forum (https://adafru.it/aHd) is a great resource for build assistance, hard-to-find parts and to show off your finished work.
- YourProps (https://adafru.it/aHe) likewise for original movie prop reference pics.
- At Maker Faire Bay Area 2012, Adam Savage (of Mythbusters fame) gave an entertaining and impassioned talk (https://adafru.it/aHf) (YouTube) about our compulsion to recreate our favorite big-screen icons.

Code

Here's the Arduino sketch (https://adafru.it/EBT) that runs the show.

```
// SPDX-FileCopyrightText: 2019 Anne Barela for Adafruit Industries
//
// SPDX-License-Identifier: MIT

// Time Circuit sketch for Adafruit 4-Digit 7-Segment Display w/I2C Backpack.
// Modeled after Doc Brown's DeLorean time circuit from the "Back to the
// Future" movies.  Uses three (3) each of the following displays:
//   Red   : http://www.adafruit.com/products/878
//   Green : http://www.adafruit.com/products/880
//   Yellow: http://www.adafruit.com/products/879
// A blue version is also available, but not used here.  Also uses two (2)
// each 3mm discrete LEDs:
```

```
//   Green : http://www.adafruit.com/products/779
//   Red   : http://www.adafruit.com/products/777
// Two yellow LEDs are used, but these aren't available in the shop -- they
// can be purchased through Digi-Key, etc.  The same sources will likely
// carry a 74HC138 3-to-8 line decoder w/inverting outputs (or, with some
// changes to the code, a more common 74HC32 quad 2-input OR gate could
// also be used, if you already have one on hand -- see comments later).
// The resulting item is not 100% screen accurate.  The film prop used
// 2-digit displays for day, hour and minute, the discrete LEDs for the
// top (red) display were yellow, while the 3-character month displays were
// back-painted glass fakes.  This demo was put together for fun, not
// pedantry, and generally gets the idea across.  Everyone recognizes it!

// These 4-digit displays can be assigned any of eight I2C addresses via
// solder jumpers on the back.  But the full time circuit requires *nine*
// displays.  A dirty hack exploits the fact that we only need one-way
// (write only) access to the displays.  The I2C data line is connected to
// all devices as normal...then the I2C clock drives the enable line of a
// 74HC138, while the select lines choose among multiple ersatz I2C buses
// (potentially up to 8, though we're only using 3 here).  Only that bus
// then responds to the incoming data.  The 74HC138 was chosen for its
// inverting outputs -- the idle state is high, consistent with I2C.

// So, each bus contains three 4-digit displays, and within each bus they're
// assigned address 0 (for MM.DD), 1 (YYYY) and 2 (HH:MM).  This simplifies
// the code, as display addresses are the same across all dates.  A ChronoDot
// (RTC_DS1307) clock chip is also used...bidirectional communication works
// because this connects to the regular I2C clock & data lines, not one of
// the 74HC138 outputs.  Altogether there's 10 devices attached to the I2C
// data line.  I'm not sure what the recommended fanout is for the ATmega...
// but if this runs into trouble, can always add a second 74HC138 for data,
// using the same bus select bits.

#include <Wire.h>
#include <Adafruit_LEDBackpack.h>
#include <Adafruit_GFX.h>
#include <RTClib.h>

Adafruit_7segment matrix[3] = {
  Adafruit_7segment(), Adafruit_7segment(), Adafruit_7segment() };
RTC_DS1307 clock;
// RTClib isn't pre-Y2K (or post-Y2.1K) compliant, so an ugly trick is used
// to represent "fantasy" dates.  Because our clock displays don't show
// seconds, that field in the DateTime class is borrowed to indicate a
// century #.  Honest-to-goodness DateTimes will have a seconds() value of
// 0 to 59.  If seconds >= 100, this field (-100) is the two-digit century.
DateTime dest(55, 11,  5, 22,  4, 100 + 19), // Nov 5, 1955 10:04 PM
         last(85, 10, 26,  1, 24, 100 + 19); // Oct 26, 1985 1:24 AM

void setup() {
  uint8_t m, b;

  clock.begin();

  // NOTE: all pin numbers used here are for the Teensy microcontroller
  // board, NOT a stock Arduino.  You may need to adapt this code to
  // your particular situation. (e.g. if making a prop that also uses
  // the Wave Shield to add sounds, you'd want to use an Arduino Uno and
  // then steer clear of all the SPI pins (10-13)).

  // Enable select lines for I2C multiplexing (only 2 are used here)
  pinMode( 9, OUTPUT);
  pinMode(10, OUTPUT);

  // Enable AM/PM LED outputs, set all LOW (off)
  for(b=11; b<=16; b++) {
    pinMode(b, OUTPUT);
    digitalWrite(b, LOW);
  }
```

```
  // Initialize all three displays on all three buses
  for(b=0; b<3; b++) {
    selectBus(b);
    for(m=0; m<3; m++) matrix[m].begin(0x70 + m);
  }
}

boolean dots = false; // For flashing colon on HH:MM times

void loop() {
  displayDate(0, dest);        // Destination time
  displayDate(1, clock.now()); // Present time
  displayDate(2, last);        // Last time departed

  dots = !dots;                // Blink blink blink
  delay(500);
}

// This function enables one of the 3 (but potentially up to 8) I2C buses
// by setting the select lines on the 74HC138 (only 2 lines are used in the
// circuit, the third is tied to ground).  A 74HC32 quad OR gate could also
// be used, but each bus will require its own select line (set corresponding
// output HIGH to disable, LOW to enable).  I wanted to use the least pins
// so that others remain free for the addition of a possible keypad later.
// (Could also free up pins using a port expander or shift register for
// the AM/PM LEDs.)
void selectBus(uint8_t b) {
  digitalWrite(10, (b & 1) ? HIGH : LOW);
  digitalWrite( 9, (b & 2) ? HIGH : LOW);
  // Can add third bit here if more buses are needed
}

// Show contents of DateTime object on the three displays across one bus
void displayDate(uint8_t b, DateTime d) {
  int x;

  selectBus(b);

  // Write MM.DD (zero padded) to display #0
  x = d.month();
  matrix[0].writeDigitNum(0, x / 10);
  matrix[0].writeDigitNum(1, x % 10, true);
  x = d.day();
  matrix[0].writeDigitNum(3, x / 10);
  matrix[0].writeDigitNum(4, x % 10);

  // Write year to display #1 (4-digit, zero-padded)
  // Great Scott!  RTClib isn't pre-Y2K compliant, so 'second'
  // is borrowed as a flag to indicate fantasy dates.
  if((x = d.second()) >= 100) x = d.year() - 2000 + x * 100;
  else                        x = d.year();
  matrix[1].writeDigitNum(0, (x / 1000) % 10);
  matrix[1].writeDigitNum(1, (x /  100) % 10);
  matrix[1].writeDigitNum(3, (x /   10) % 10);
  matrix[1].writeDigitNum(4,  x         % 10);

  // Write time to display #2 (HH:MM, zero-padded + AM/PM indicator)
  x = d.hour();
  if(x < 12) { // AM
    digitalWrite(b * 2 + 11, HIGH); // Upper LED on
    digitalWrite(b * 2 + 12, LOW);  // Lower LED off
  } else {     // PM
    digitalWrite(b * 2 + 11, LOW);  // Upper LED off
    digitalWrite(b * 2 + 12, HIGH); // Lower LED on
  }
  if(x > 12) x -= 12; // Convert 24-hour to 12-hour time
  matrix[2].writeDigitNum(0, x / 10);
  matrix[2].writeDigitNum(1, x % 10);
```

```
  x = d.minute();
  matrix[2].writeDigitNum(3, x / 10);
  matrix[2].writeDigitNum(4, x % 10);
  matrix[2].drawColon(dots);

  // Refresh all three displays on current bus
  for(x=0; x<3; x++) matrix[x].writeDisplay();
}
```