Tap Tempo Trinket

Created by Phillip Burgess

https://learn.adafruit.com/tap-tempo-trinket
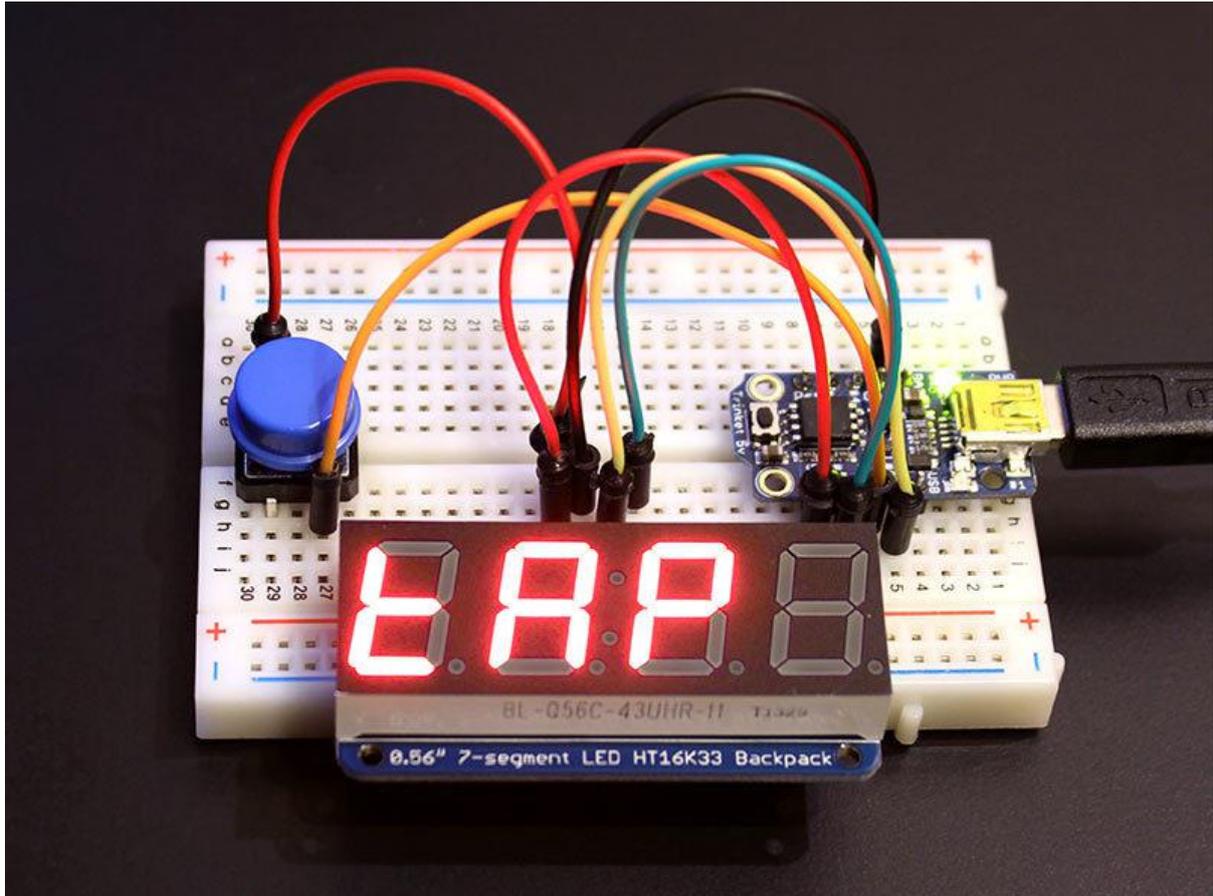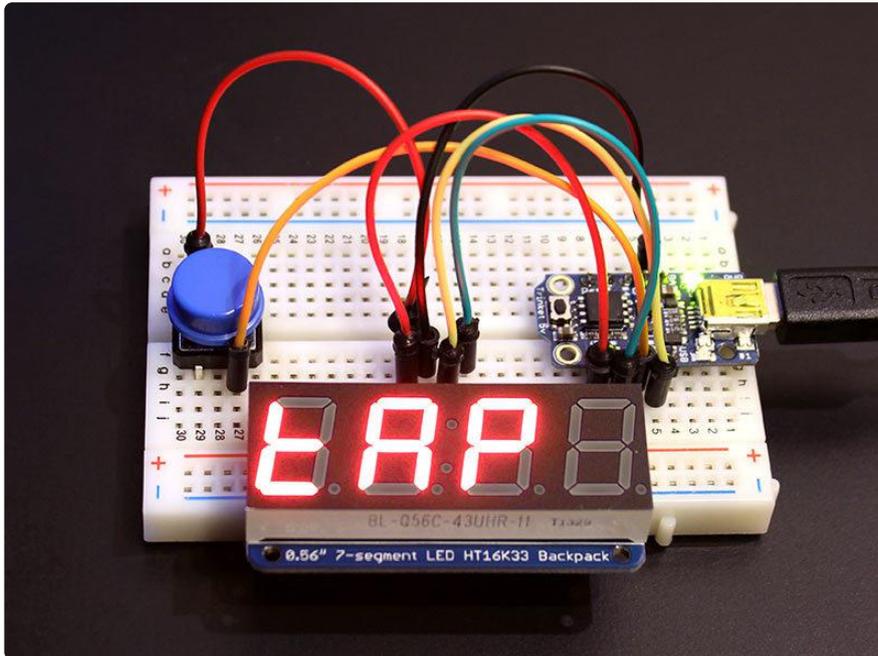
Last updated on 2023-08-29 02:23:16 PM EDT

Table of Contents

Overview

This beats-per-minute calculator is a quick and easy project using the Adafruit Trinket microcontroller board and 7-segment LED Backpack.

You: tap the button in time with music.
Trinket: reports the corresponding beats-per-minute.

Parts you'll need:

- Adafruit Trinket (either the 5V (http://adafru.it/1501) or 3.3V (http://adafru.it/1500) variety)
- 7-segment LED Display Backpack (any color, 0.56" size, not 1.2")
- Momentary "normally open" pushbutton
- Breadboard, jumper wires, USB cable (A to mini B)

If this is your first time using Trinket, work through the Introducing Trinket () guide first; you need to customize some settings in the Arduino IDE. Once you have it up and running, you'll then install the following libraries:

- Adafruit_LEDBackpack ()
- Adafruit_GFX () and Adafruit_BusIO () (required by Adafruit_LEDBackpack, even though we're not doing graphics)

Installing Arduino libraries is a frequent stumbling block. If this is your first time, or simply needing a refresher, please read the All About Arduino Libraries () tutorial.

Schematic

If using a solderless breadboard, first install row pin headers on both your Trinket board and the LED Backpack. The backpack requires soldering the LED display to the board…follow the Adafruit LED Backpacks () guide and make sure it's lined up the right way!

To assemble the project, only six connections are needed:
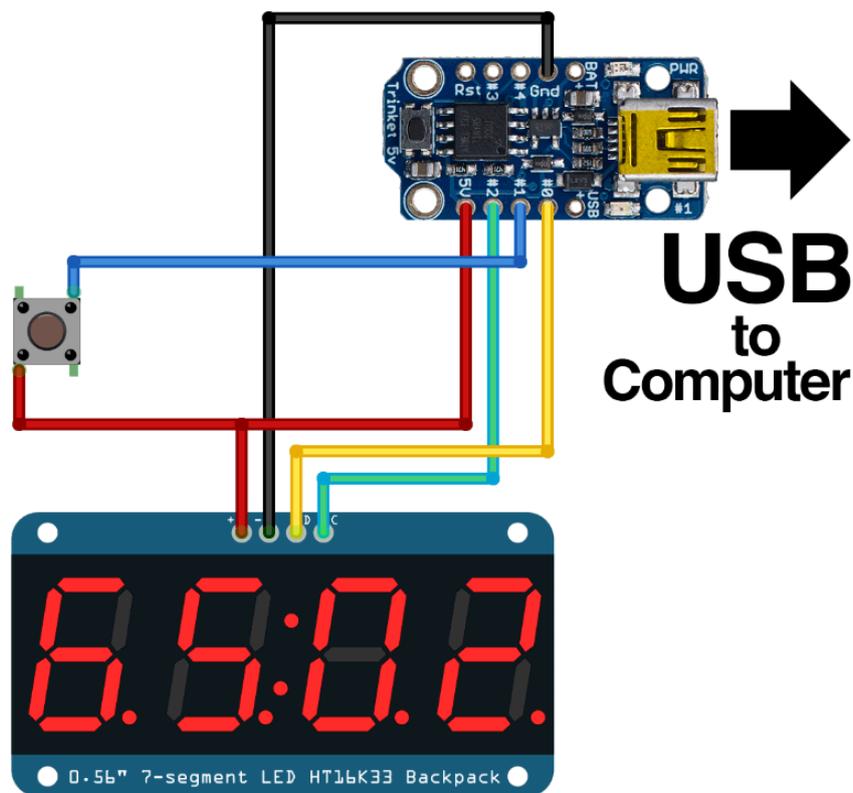
The diagram shows a 5V Trinket board, but the project is interchangeable with the 3.3V version; everything is wired the same.

- Connect Trinket's 5V (or 3V) pin to + on the LED backpack and one leg of the pushbutton.
- Connect the opposite leg of the pushbutton to Trinket pin #1.
- Connect Trinket's GND pin to – on the backpack.
- Connect Trinket Pin #0 to D (data) on the backpack.
- Connect Trinket Pin #2 to C (clock) on the backpack.

Unlike some regular Arduino projects, the choice of pins here is not negotiable. The selection will be explained later.

For simplicity's sake we're using the USB connection for power. For portable use, an optional battery pack can be connected to the Trinket's BAT and GND pins.

Code

The following code can be copied-and-pasted into a new project window in the Arduino IDE.

From the Tools→Board menu, select Adafruit Trinket 8 MHz (3.3V or 5V boards) or Trinket 16 MHz (5V board only). Connect the USB cable between the computer and Trinket, press the reset button on the board, then click the upload button (right arrow icon) in the Arduino IDE.

If you get an error message (or a huge list of them), it's usually one of the following:

- Is the Arduino IDE properly configured for Trinket use? Try compiling and uploading a simple sketch (like the Blink example, set for pin #1). From the Tools→Programmer menu, "USBtinyISP" should be selected.
- Are the libraries properly installed? They must be correctly named and in the right location. If you already had one or more of these libraries previously installed, download the latest versions — some changes may have been made specifically for Trinket. (Same goes for Boards manager support...a recent change for Trinket removed the need for a separate TinyWireM library, so make sure the board definitions are current.)
- If the code compiles but doesn't upload, press the reset button and try again.

```
/*******************************************************************
  Tap tempo sketch for Adafruit Trinket.  Tap a button in sync with
  a beat, LED display shows beats per minute.  Stop tapping for
  4 second to reset counter.

  Required libraries include Adafruit_LEDBackpack and Adafruit_GFX.

  Required hardware includes an Adafruit Trinket microcontroller
  (5V or 3.3V), an Adafruit 7-Segment LED Backpack, and a momentary,
  normally-open pushbutton.

  As written, this is specifically for the Trinket; would need
  modifications on other boards (Arduino Uno, etc.).

  Trinket :   5V/3V   Gnd   Pin #0   Pin #2
  Backpack:     +      -       D        C

  Pushbutton between Pin #1 and 5V/3V.
  *****************************************************************/

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

```
#include <avr/power.h>

Adafruit_7segment disp = Adafruit_7segment();

unsigned long
  prevBeat = 0L, // Time of last button tap
  sum      = 0L; // Cumulative time of all beats
uint16_t
  nBeats   = 0;  // Number of beats counted
uint8_t
  prevButton;    // Value of last digitalRead()

void setup() {
  if(F_CPU == 16000000) clock_prescale_set(clock_div_1);
  disp.begin(0x70);
  prevButton = digitalRead(1);
}

static unsigned long debounce() { // Waits for change in button state
  uint8_t        b;
  unsigned long start, last;
  long           d;

  start = micros();

  for(;;) {
    last = micros();
    while((b = digitalRead(1)) != prevButton) { // Button changed?
      if((micros() - last) > 25000L) {          // Sustained > 25 mS?
        prevButton = b;                         // Save new state
        return last;                            // Return time of change
      }
    } // Else button unchanged...do other things...

    d = (last - start) - 4000000L; // Function start time minus 4 sec
    if(d > 0) {                          // > 4 sec with no button change?
      nBeats = 0;                 // Reset counters
      prevBeat = sum = 0L;
    }

    // If no prior tap has been registered, program is waiting
    // for initial tap.  Show instructions on display.
    if(!prevBeat) {
      if(!(d & 0x00100000)) { // ~1.05 second cycle
        disp.writeDigitRaw(0, 0x78); // t
        disp.writeDigitRaw(1, 0x77); // A
        disp.writeDigitRaw(3, 0x73); // P
        disp.writeDigitRaw(4, 0x00);
      } else {
        disp.writeDigitRaw(0, 0x7C); // b
        disp.writeDigitRaw(1, 0x79); // E
        disp.writeDigitRaw(3, 0x77); // A
        disp.writeDigitRaw(4, 0x78); // t
      }
      disp.writeDisplay();
    }
  }
}

void loop() {
  unsigned long t;
  uint16_t      b;

  t = debounce(); // Wait for new button state

  if(prevButton == HIGH) {              // Low-to-high (button tap)?
    if(prevBeat) {                      // Button tapped before?
      nBeats++;
      sum += 600000000L / (t - prevBeat); // BPM * 10
```

```
    b    = (sum / nBeats);            // Average time per tap
    if(b &gt; 9999) b = 9999;
    disp.print(b);
    disp.writeDigitNum(3, (b/10)%10, true); // .
  } else {                            // First tap
    disp.clear();                     // Clear display, but...
    disp.writeDigitRaw(3, 0x80);      // a dot shows it's on
  }
  disp.writeDisplay();
  prevBeat = t;                       // Record time of last tap
  }
}
```

There's not much to it! This is the sort of project where Trinket really shines…a small, focused project where a full-blown Arduino would just be wasted potential.

The largest section of the code (aside from the comments up top) is simply to flash "TAP BEAT" at startup. Button input is debounced, time between button taps is then calculated using the micros() timer, and BPM is figured by dividing 600,000,000 (10X the number of microseconds in 1 minute) by this time interval. The 10X figure is just so we can look extra geeky by then adding a decimal point.

Things Learned

Trinket projects have their own unique challenges. It's not simply the matter of less RAM, code space and fewer pins. In such a constrained system, the interactions between parts sometimes require careful planning and working in unusual ways…

- To simplify the parts list, we're using the USB connection for power. Trinket pins #3 and #4 are involved in USB communication. Even when we're not uploading to the chip, there may be other "chatter" on the bus…and using these pins as inputs would cause false readings. An alternative design to regain use of these pins would be to disconnect the board from USB after programming and attach a battery to the BAT and GND pins. But these same pins are usually fine for outputs and won't interfere.
- Pins #0 and #2 are reserved when using I^2C communication (via the Wire library), required for the LED backpack in this project.
- That leaves pin #1 as the last possible input…but this pin is also connected to the board's red status LED. The usual technique for connecting a button — using the microcontroller's internal pull-up resistors — won't work here (the LED turns on and always pulls the input low). So we cheat a little here and use the same LED (and its associated resistor) as a pull-down. One leg of the pushbutton connects to this pin, the other leg to the +5V/3V supply. The pin then reads as HIGH when the button is pressed, LOW when released. This has a nifty secondary bonus effect that the LED lights each time the button is pressed. (Pin #1 is fine as an input as long as you don't require the internal pull-up.)

While the 7-segment LED backpack works with Trinket, our 8x8 LED matrix backpacks do not. There simply isn't enough room for the code. Yet we still need to #include the Adafruit_GFX library...the LED backpack library depends on it (only the 7-segment parts are linked into our sketch, that's how it manages to fit). That's not to say that Trinket couldn't control the matrices...but it would require developing a smaller, limited library for that one task.