Trinket Powered Analog Meter Clock

Created by Anne Barela

https://learn.adafruit.com/trinket-powered-analog-meter-clock

Last updated on 2024-06-03 01:24:13 PM EDT

Table of Contents

Overview

This guide is written for the original Adafruit Trinket board using an ATtiny85 microcontroller. It uses features specific to that chip including the timers, which is different from other microcontrollers. If you are using another microcontroller, please search the web for a more modern guide.

Trinket lends itself very well to building clock projects, its small and easy to hide behind a larger display. And clocks don't need a lot of logic, this example only has maybe 20 lines of code. Adding a digital display via I2C is possible using seven segment or character-based displays (with the library code posted for other projects).

This project interfaces Trinket to the the Adafruit DS1307 real-time clock (RTC) breakout board to form a clock. But rather than use the traditional digital display, the display is done using two analog meters. One for hours, one for minutes.

The Trinket can output to a meter without digital to analog converters. Trinket has pulse width modulation (PWM) on three of its pins. The meter uses a moving coil inductance movement, acting to average the indication of current flowing through it. If you have narrow pulses, the average voltage it sees is lower, thus the current is lower for the fixed resistance attached to it. For wide pulses, the meter sees nearly the supply voltage and will stay around the full scale. This circuit varies the pulse width sent to the meters proportional to the hour of the day and the minutes after the hour.

For two meters, we will use two of the three PWM pins on Trinket (the third is also an I2C pin connected to the clock module).

There are several projects on the web using analog meters to tell time. The ease at which you can do this with Trinket allows you to build this type of clock quickly and compactly. You may focus on designing how to mount the meters in a creative way.

There are many ways to display the finished project. Rather than a cabinet or plexiglass display, I chose meters free-floating in a colorful box. I think it lends a modern look.

You can check Google Images for "Clock Analog Meter" for other mounting designs. Designing your own solution is the best part of such a project.

Building the Circuit

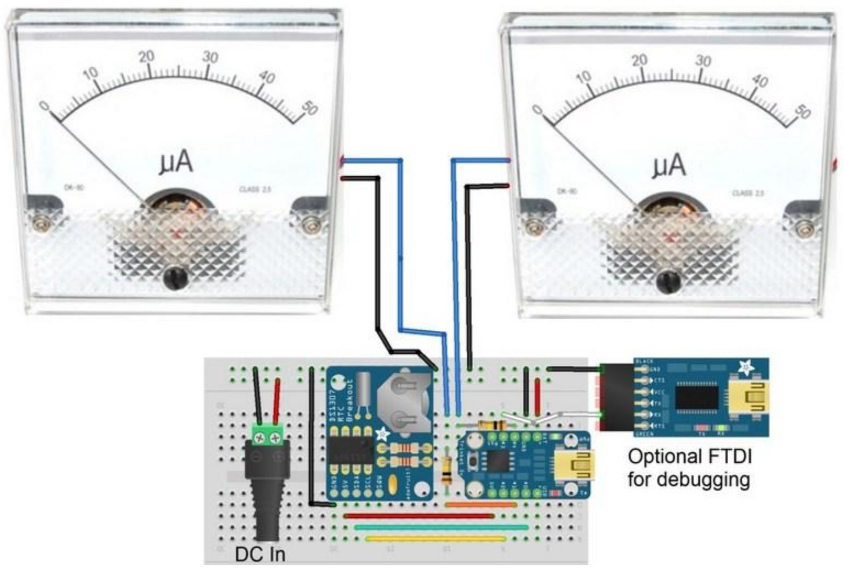
Start by unpacking your Trinket. If you will use a breadboard or Perma-Proto board, you will want to solder on the header pins (provided). See Introducing Trinket (https://adafru.it/cEu) on doing this and general information.

Unpack your DS1307 kit. This requires assembly also. Please follow the DS1307 Real Time Clock Breakout Board Kit tutorial (https://adafru.it/cO8) on building your clock module.

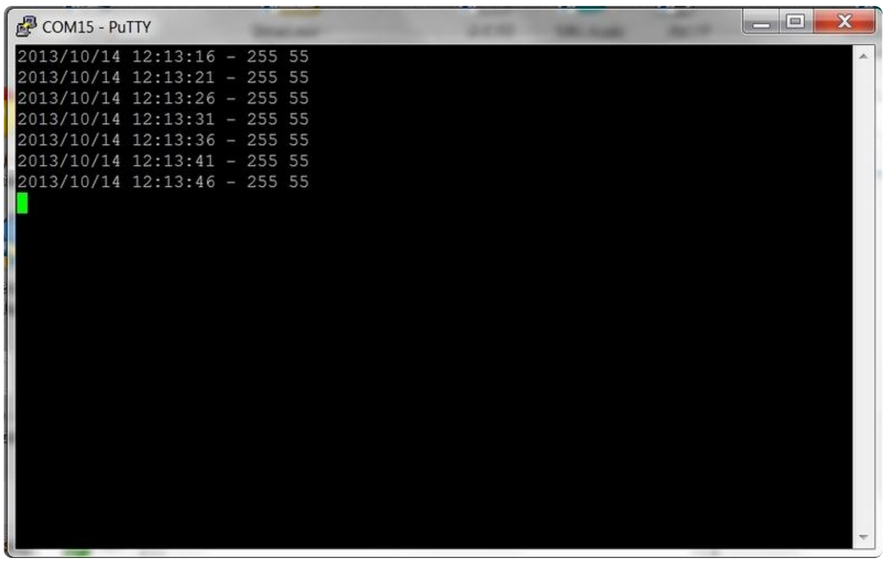
Trinket can be powered from 3.7 to 16 volts via the BAT+ input and ground. This makes powering the clock very flexible. For this project, I chose the 5V Trinket as the DS1307 board has a 5 volt input which may be connected to the 5V output pin on Trinket. If you use another RTC module that works at 3.3 volts, the Trinket 3V may be used with appropriate changes to the meter calibration. I show powering via DC supply (wall wart). Battery use will vary depending on the batteries you choose. A 9 volt battery will not last very long and is not recommended.

For a 5 volt Trinket and 50 microamp meters, for full scale deflection we need a series resistor on each meter to keep the current less than or equal to the maximum current the meter can handle. Using Ohm's law, R = V / I = 5 / .00005 = 100,000 ohms (100 K). You will need two of these resistors, preferably 5% or better tolerance. These are commonly available from electronics suppliers. If you want precision in calibrating the meter, you may want to substitute each resistor with a 100K potentiometer with a series resistor, perhaps 10 to 47 K. This allows for tuning the resistance. When I designed the project, the 100K resistors gave accurate enough time without needing potentiometers.

Do not directly connect the meter to a source of voltage as it will damage the meter. Use an appropriate series resistor in the circuit to limit the meter current.



Wiring is straightforward. All the pins are used except GPIO #3. I used this pin temporarily to connect to an FTDI Friend. On Trinket you can run a software serial library that transmits only. This was handy in debugging the circuit as it gives you console-like output using only one pin and ground.



```
2013/10/14 12:13:16 - 255 55
2013/10/14 12:13:21 - 255 55
2013/10/14 12:13:26 - 255 55
2013/10/14 12:13:31 - 255 55
2013/10/14 12:13:36 - 255 55
2013/10/14 12:13:41 - 255 55
2013/10/14 12:13:46 - 255 55
```

The text will show the date and time along with two numbers which represent numbers from 0 to 255 for a pulse width corresponding to the time. Above, 255 shows this is noon, 13 minutes after the hour is 55/255 (not quite 1/4). If you are not getting this type of output on serial, check your connections and code.

Preparing to Code

Using the Introducing Trinket (https://adafru.it/cN0) tutorialto set up the Arduino integrated development environment (IDE) on your system and add Adafruit AVR board support (which includes Trinket) to your IDE.

Follow the instructions in the Introducing Trinket tutorial to set up your Arduino development environment.

When you load a program, you must press the hardware reset button on the Trinket then quickly press upload in the Arduino software to upload a sketch. If you get an error, try the reset-upload process again. If you continually cannot load the blink sketch, check to make sure the Trinket is connected (without any wires connected to pins #3 and #4) and the Arduino IDE software has all the required changes.

For preparing the Arduino IDE for the clock program, we need a software library to access the real time clock. We'll be using the following Arduino libraries:

1. The built-in Arduino I2C Wire library.
2. The Adafruit RTClib library (https://adafru.it/c7r)

Here’s a tutorial (https://adafru.it/aYM) that walks through the process of correctly installing Arduino libraries.

Now you are ready to copy the sketch on the next page for your clock.

Debugging Issues

For errors in the Arduino IDE software:

- Ensure you have installed the Adafruit AVR boards support and have selected Trinket 8 MHz as the board.
- Ensure you have installed the RTClib library downloaded fresh from the Adafruit GitHub site.
- Ensure you push the Trinket on board reset button before uploading your sketch, the red LED will blink when ready for upload, there is a 10 second window to do this.

- If you place a large amount of code or other libraries in a sketch, it is very easy to exceed the available code space on the Trinket. If your program absolutely will not fit, consider switching to an Arduino Uno (http://adafru.it/50), Adafruit Boarduino (https://adafru.it/cKj), Pro Trinket (http://adafru.it/2000), or Adafruit Flora (https://adafru.it/aSZ) with standard libraries.
- If you get errors similar to the one below, you may have included decimal numbers and the floating point library was added by the Arduino IDE, exceeding the amount of program space available.

arduino-1.0.1/hardware/tools/avr/bin/../lib/gcc/avr/4.3.2/../../../../avr/lib/**avr25**/crttn85.o:(.init9+0x2):**relocation truncated to fit: R_AVR_13_PCREL** against symbol `exit' defined in .fini9 section in /arduino-1.0.1/hardware/tools/avr/bin/../lib/gcc/avr/4.3.2/**avr25**\libgcc.a(_exit.o)

The code on the next page still leaves a good deal of space for additional functionality, especially once the serial code is no longer needed. If you declare large arrays, define large text strings, add decimal/floating point numbers, or other libraries, the space may fill up or overflow.

Code

You will want to run the sketch twice, once to set the clock, another to have it operate. The code in setup() checking for rtc.isrunning() should be uncommented. This will set the clock to the time your code is compiled. You can then recomment out that code as the DS1307 will keep the time.

If you plan to have the code function differently than the sample or have problems you want to debug, having a serial monitor will help. I have tried the SendOnlySoftwareSerial library available in this arduino.cc post (https://adafru.it/cOa) with good results. It adds about 1300 bytes of overhead so you will want to comment out the serial code in your sketch when you do not need it.

```
// Adafruit Trinket analog meter clock
// Date and time functions using a DS1307 RTC connected via I2C and the Wire lib
//
// Version 2.0 February 2016 to use new Arduino 1.6.7+ IDE and Wire library
//             Mike Barela for Adafruit Industries

// Download the RTClib library from Adafruit's Github repository and
//    install in your Arduino Libraries directory
#include &lt;RTClib.h&gt;

//For debug, uncomment serial code, use a FTDI Friend with its RX pin connected to 
Pin 3
//   You will need a terminal program (such as freeware PuTTY for Windows) set to 
the
//   USB port of the FTDI friend at 9600 baud.  Uncomment out Serial commands to
```

```
see what's up
//#include &lt;SendOnlySoftwareSerial.h&gt;  // See http://forum.arduino.cc/
index.php?topic=112013.0

#define HOUR_PIN     1   // Hour display via PWM on Trinket GPIO #1
#define MINUTE_PIN   4   // Minute display via PWM on Trinket GPIO #4 (via Timer 1
calls)

//SendOnlySoftwareSerial Serial(3);  // Serial transmission on Trinket Pin 3
RTC_DS1307 rtc;                      // Set up real time clock

void setup () {
  pinMode(HOUR_PIN, OUTPUT);    // define PWM meter pins as outputs
  pinMode(MINUTE_PIN, OUTPUT);
  PWM4_init();                  // Set timer 1 to work PWM on Trinket Pin 4

  rtc.begin();                  // Begin DS1307 real time clock
  //Serial.begin(9600);           // Begin Serial Monitor at 9600 baud
  if (! rtc.isrunning()) {
    //Serial.println("RTC is NOT running!");
    // following line sets the RTC to the date &amp; time this sketch was compiled
    //rtc.adjust(DateTime(__DATE__, __TIME__));
  }
}

void loop () {
    uint8_t hourvalue, minutevalue;
    uint8_t hourvoltage, minutevoltage;

    DateTime now = rtc.now();           // Get the RTC info
    hourvalue = now.hour();             // Get the hour
    if(hourvalue &gt; 12) hourvalue -= 12; // This clock is 12 hour, is 13-24,
convert to 1-12
    minutevalue = now.minute();         // Get the minutes
// if you have calibration issues, you can change the last two values (zero higher,
255 lower)
// to have the needle move less if your scale is not pasted on 100% straight or
// if you decide to use different meters from the Adafruit products.
    hourvoltage = map(hourvalue, 0, 12, 0, 255);     // Convert hour to PWM duty
cycle
    minutevoltage = map(minutevalue, 0, 60, 0, 255); // Convert minutes to PWM duty
cycle
/*
    // Uncomment out this and other serial code to check that your clock is
working.
    Serial.print(now.year(), DEC);
    Serial.print('/');
    Serial.print(now.month(), DEC);
    Serial.print('/');
    Serial.print(now.day(), DEC);
    Serial.print(' ');
    Serial.print(now.hour(), DEC);
    Serial.print(':');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    Serial.print(now.second(), DEC);
    Serial.print(" - ");
    Serial.print(hourvoltage, DEC);
    Serial.print(' ');
    Serial.print(minutevoltage, DEC);
    Serial.println();
*/
    analogWrite(HOUR_PIN, hourvoltage);
    analogWrite4(minutevoltage);

    // code to put the processor to sleep might be preferable - we will delay
    delay(5000);  // check time every 5 seconds.  You can change this.
}
```

```
void PWM4_init() {
  // Set up PWM on Trinket GPIO #4 (PB4, pin 3) using Timer 1
  TCCR1 = _BV (CS10);             // no prescaler
  GTCCR = _BV (COM1B1) | _BV (PWM1B);  //  clear OC1B on compare
  OCR1B = 127;                    // duty cycle initialize to 50%
  OCR1C = 255;                    // frequency
}

// Function to allow analogWrite on Trinket GPIO #4
void analogWrite4(uint8_t duty_value) {
  OCR1B = duty_value;  // duty may be 0 to 255 (0 to 100%)
}
```

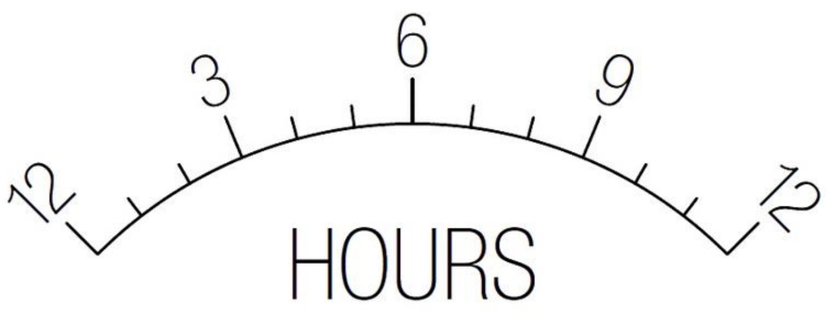
PWM on Pin 4

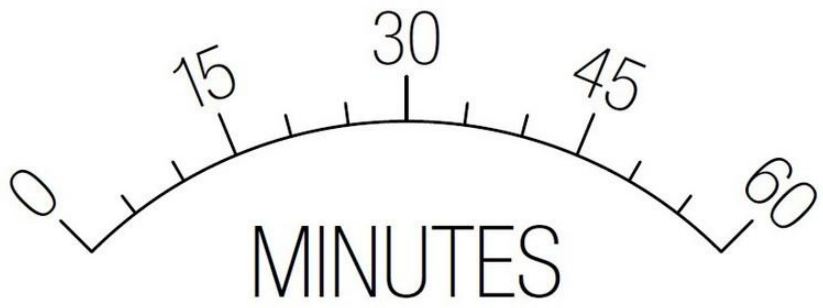
Trinket tinkers have noted that the analogWrite function will initiate pulse width modulation on pins #0 and #1 but not pin #4 (although the pin is PWM capable). The Arduino IDE does not set it up seamlessly. Function PWM4_init in the code above sets up the ATTiny85 Timer 1 to provide PWM at 50% duty cycle initially. Calls to the new function analogWrite4 will vary the PWM as desired. Using Timer 1 for PWM takes away its use for other things but for the clock project, it is not needed elsewhere. Just be sure some other code or library is not expecting to use it. This has now been posted to Google+ and the Adafruit Trinket forum (https://adafru.it/cOb) for other uses.

Setup and Mounting

Setup

You will want to change the meter faces to have them display hours and minutes instead of microamperes, Two basic designs are below. There are other creative designs on the Internet.

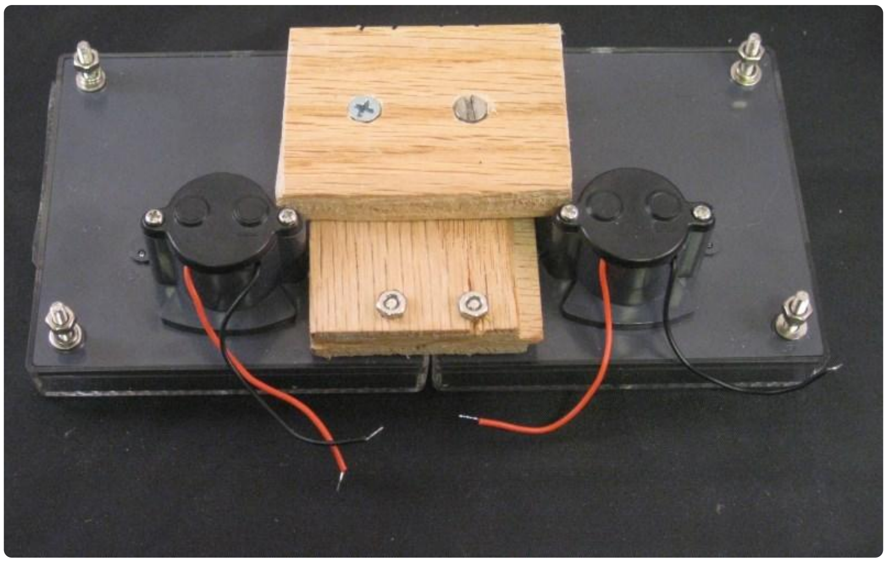




Carefully remove the two silver screws on either side of a meter. Lift up on the cover. Cut your meter face out of paper. Ensure at the bottom you cut a semicircle out so the needle movement will swing freely. Use a glue stick or other very light adhesive on the meter face then carefully slide the new face in without harming the meter needle which is very fragile. You will wish to make slight adjustments to align the scale. The meter needle should be pointing at the left hand mark on the scale (if a tiny bit off, you can adjust this later). Put the cover back on the meter and screw it on. Using a flat screwdriver, you can adjust the zero on the meter slightly with the black screw in the lower middle of the meter.

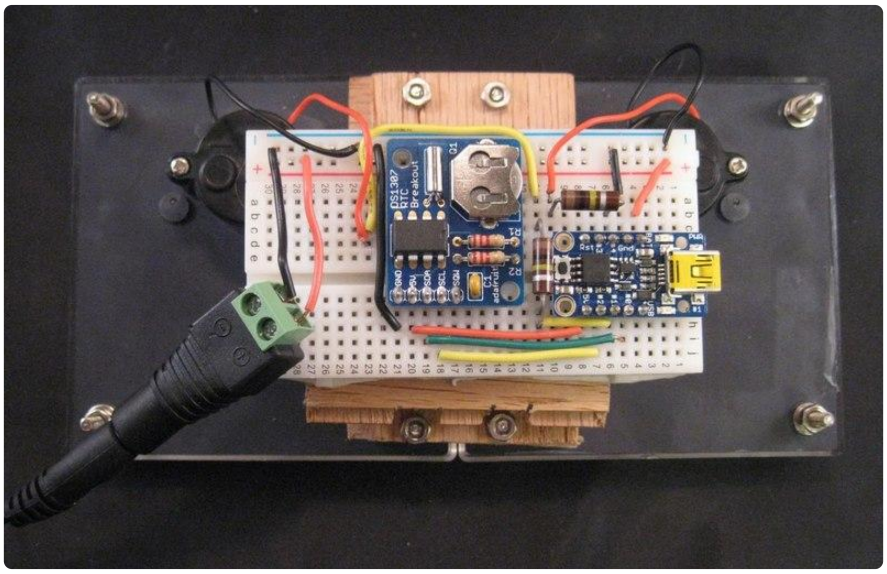
Mounting

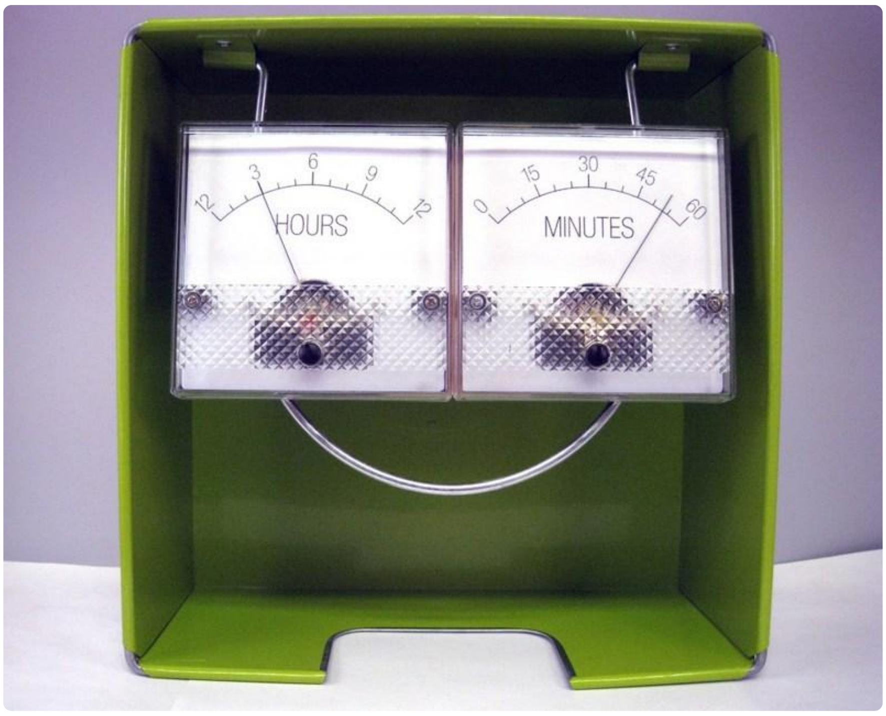
The meters have four mounting posts providing a sturdy mount on nearly any surface. For my less than traditional mounting method, I placed the meters side by side. I planned to put the circuit board behind the meters. For this I nned a flat surface and the meter movements stick out from the rear. I fashioned 3/8" thick wood precut to 1 3/4" widths. I cut one piece at 2 7/8" to connect the meters, another at 2 1/4" to bring the level up to the back of the meter movements. Two screws (not too long!) from the screw bin were inset to connect the pieces and it was mounted to the meter with the included nuts. A 3D printed mount would be another method to make custom pieces.



The circuit board is mounted on the back of the meters in my design. I am mounting the breadboard, I highly suggest when you are satisfied with the circuit, you transfer it to a Perma-Proto board to provide sturdy, permanent connections. You will want to secure the 9 volt or other battery such that you can change the batteries easily. You may design your clock with a DC wall supply to avoid batteries, the trade-off being having to connect it to wall current. The DS1307 board will continue to hold the time for many months with no power due to the on-board coin-cell battery.

If you mount the meters in a box, you have much more flexibility on circuit board placement.





Post your designs in the Adafruit Trinket forum (https://adafru.it/cOb) or the Adafruit Google+ Makers, hackers, artists & engineers community (https://adafru.it/cKZ).

And the modern mounting box I used. I borrowed the napkin holder bought at clearance at Kohls. It's not what you get, it's what you do with it.