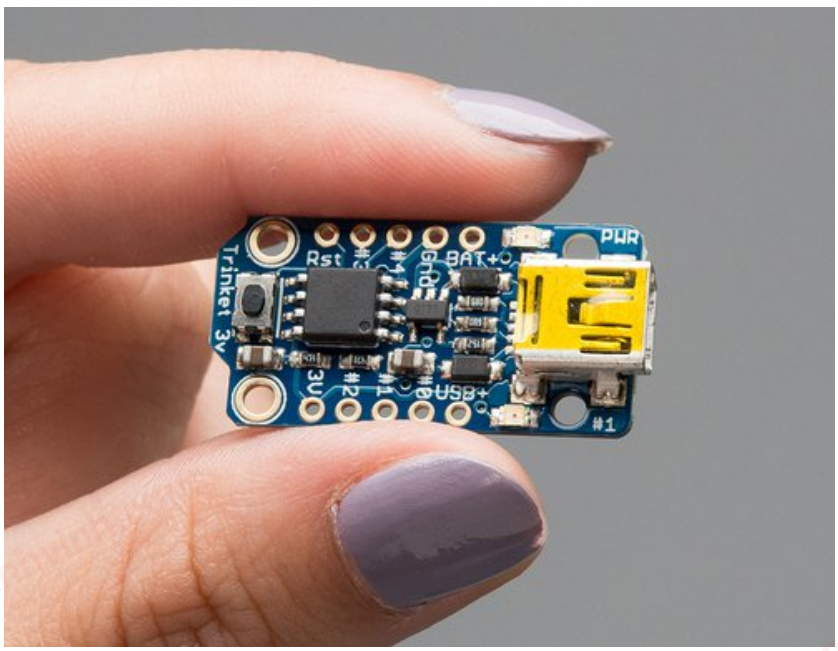
chanhvinh.com

Introducing Trinket

Created by lady ada



Last updated on 2016-09-09 07:45:55 PM UTC

chanhvinh.com

Guide Contents

Guide Contents 2
Introduction 4
Guided Tour 7
Pinouts 9
Power Pins 11
GPIO Pins 11
USB Pins 12
Reset and Regulator Output 12
Starting the Bootloader 14
About the bootloader 14
Trinket USB Drivers for Windows 14
Special Notes on using Trinket with Linux 14
How to start the bootloader 15
Windows Driver Installation 16
Setting up with Arduino IDE 19
Arduino IDE Setup 19
Blink! 19
Something Went Wrong! 22
- If you get the error message avrdude: Error: Could not find USBtiny device (0x1781/0xc9f) 22
- If you get a lot of red text, errors and also a warning about Verification Failed 23
- On Linux if you get the error message "usbtiny_receive: error sending control message: Protocol error (expected 4, got -71)" 24

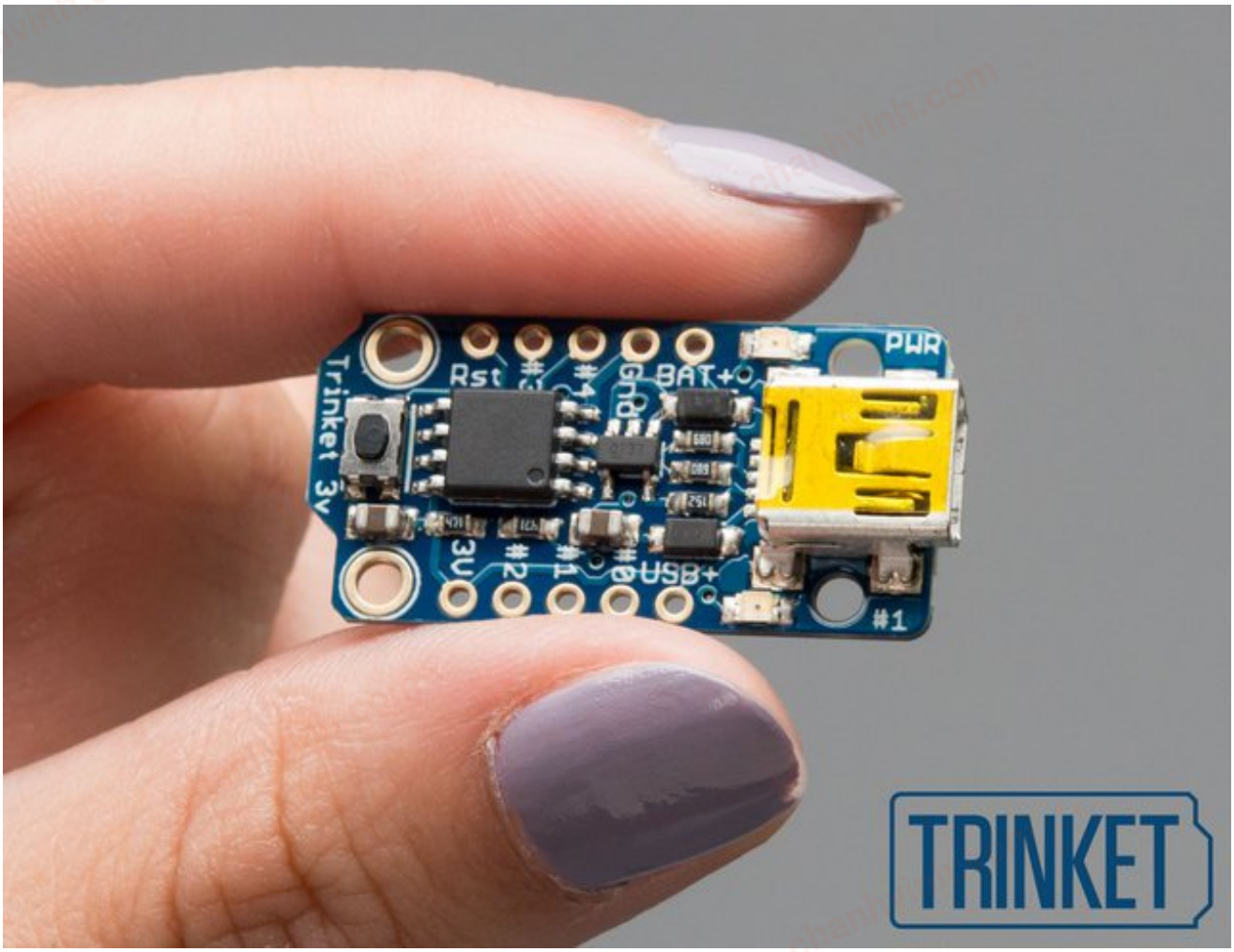
Programming with Arduino IDE 26
- pinMode() & digitalWrite() & digitalRead() 26

analogRead() 27
analogWrite() 27
Wire (i2c) 28
More... 29
Programming with AVRdude 30
The Short Way 30
The Long Way 31

chanhvinh.com

Uploading with AVRdude 37
Programming in a Blink example 38
16MHz vs 8MHz Clock 40
Power Tradeoffs 40
How to activate the 16 MHz clock 40
...on AVR-GCC 40
...Arduino IDE 41
Repairing bootloader 42
Downloads 44
Datasheets 44
Source code 44
Schematics 44
Newer Trinket (Micro USB) dimensions 46
Older Trinket (Mini USB) dimensions 47
FAQ 49

Introduction



Trinket may be small, but do not be fooled by its size! It's a tiny microcontroller board, built around the Atmel ATtiny85, a little chip with a lot of power. We wanted to design a microcontroller board that was small enough to fit into any project, and low cost enough to use without hesitation. Perfect for when you don't want to give up your expensive dev-board and you aren't willing to take apart the project you worked so hard to design. It's our lowest-cost arduino-IDE programmable board!

The Attiny85 is a fun processor because despite being so small, it has 8K of flash, and 5 I/O pins, including analog inputs and PWM 'analog' outputs. We designed a USB bootloader so you can plug it into any computer and reprogram it over a USB port just like an Arduino. In fact we even made some simple modifications to the Arduino IDE so that it works like a mini-Arduino board. You can't stack a big shield on it but for many small & simple projects

the Trinket will be your go-to platform.

Even though you can program Trinket using the Arduino IDE, it's not a fully 100% Arduino-compatible. There are some things you trade off for such a small and low cost microcontroller!

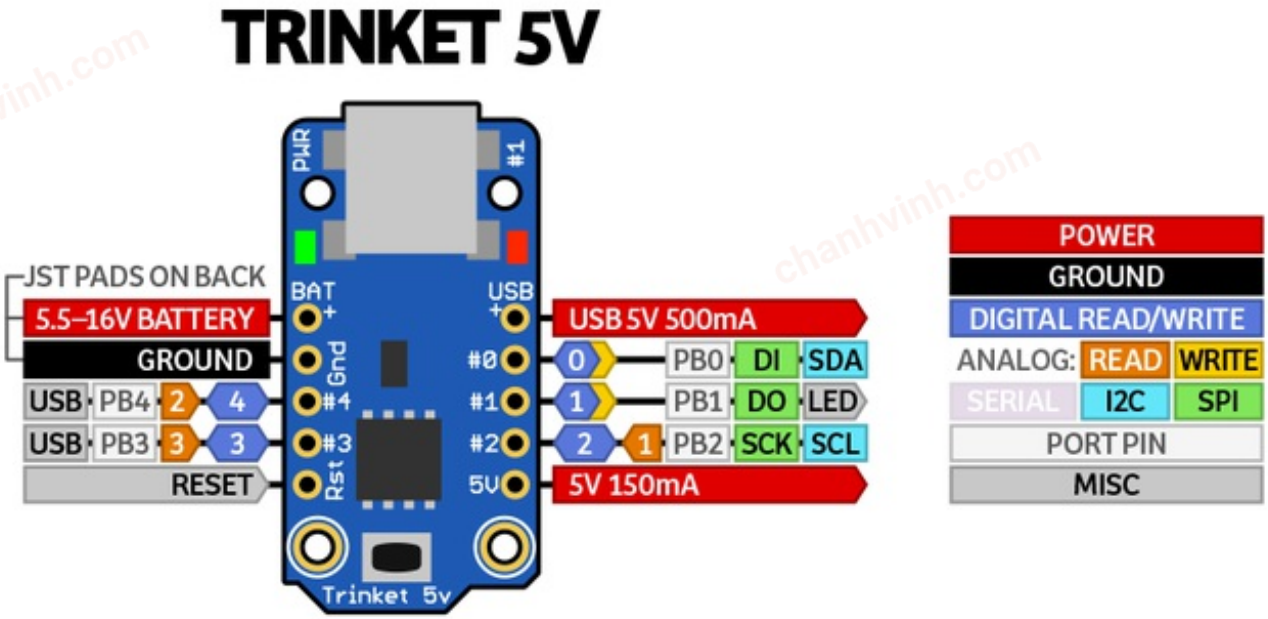
- Trinket does not have a Serial port connection for debugging so the serial port monitor will not be able to send/receive data
- Some computers' USB v3 ports don't recognize the Trinket's bootloader. Simply use a USB v2 port or a USB hub in between

There are two versions of the Trinket. One is 3V and one is 5V. Both work the same, but have different operating logic voltages. Use the 3V one to interface with sensors and devices that need 3V logic, or when you want to power it off of a LiPo battery. The 3V version should only run at 8 MHz. Use the 5V one for sensors and components that can use or require 5V logic. The 5V version can run at 8 MHz or at 16MHz by setting the software-set clock frequency.

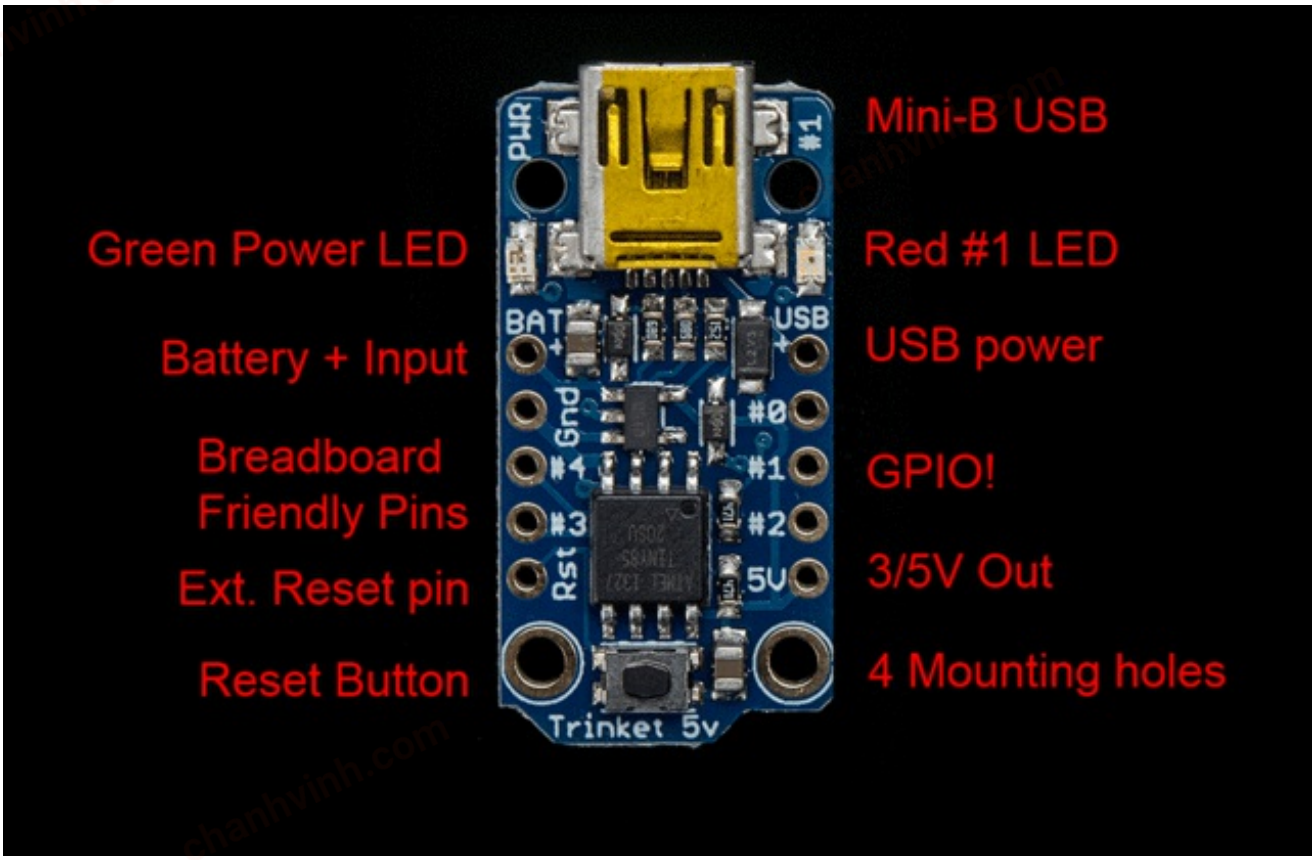
Here are some useful specifications!

- ATtiny85 on-board, 8K of flash, 512 byte of SRAM, 512 bytes of EEPROM
- Internal oscillator runs at 8MHz, but can be doubled in software for 16MHz
- **USB bootloader with a nice LED indicator looks just like a USBtinyISP so you can program it with AVRdude** (with a simple config modification) **and/or the Arduino IDE** (with a few simple config modifications)
- Mini or Micro-B USB jack for power and/or USB uploading, you can put it in a box or tape it up and use any USB cable for when you want to reprogram.
- We really worked hard on the bootloader process to make it rugged and foolproof, this board wont up and die on you in the middle of a project!
- ~5.25K bytes available for use (2.75K taken for the bootloader)
- **Available in both 3V and 5V flavors**
- On-board 3.3V or 5.0V power regulator with 150mA output capability and ultra-low dropout. Up to 16V input, reverse-polarity protection, thermal and current-limit protection.
- Power with either USB or external output (such as a battery) - it'll automatically switch over
- On-board green power LED and red pin #1 LED
- **Reset button for entering the bootloader or restarting the program. No need to unplug/replug the board every time you want to reset or update!**
- 5 GPIO - 2 shared with the USB interface. The 3 independent IO pins have 1 analog input and 2 PWM output as well. The 2 shared IO pins have 2 more analog inputs and one more PWM output.

- Hardware I2C / SPI capability for breakout & sensor interfacing.
- **Mounting holes! Yeah!**



Guided Tour



Let me take you on a tour of your Trinket! Each trinket is assembled here at Adafruit and comes chock-full of good design to make it a joy to use.

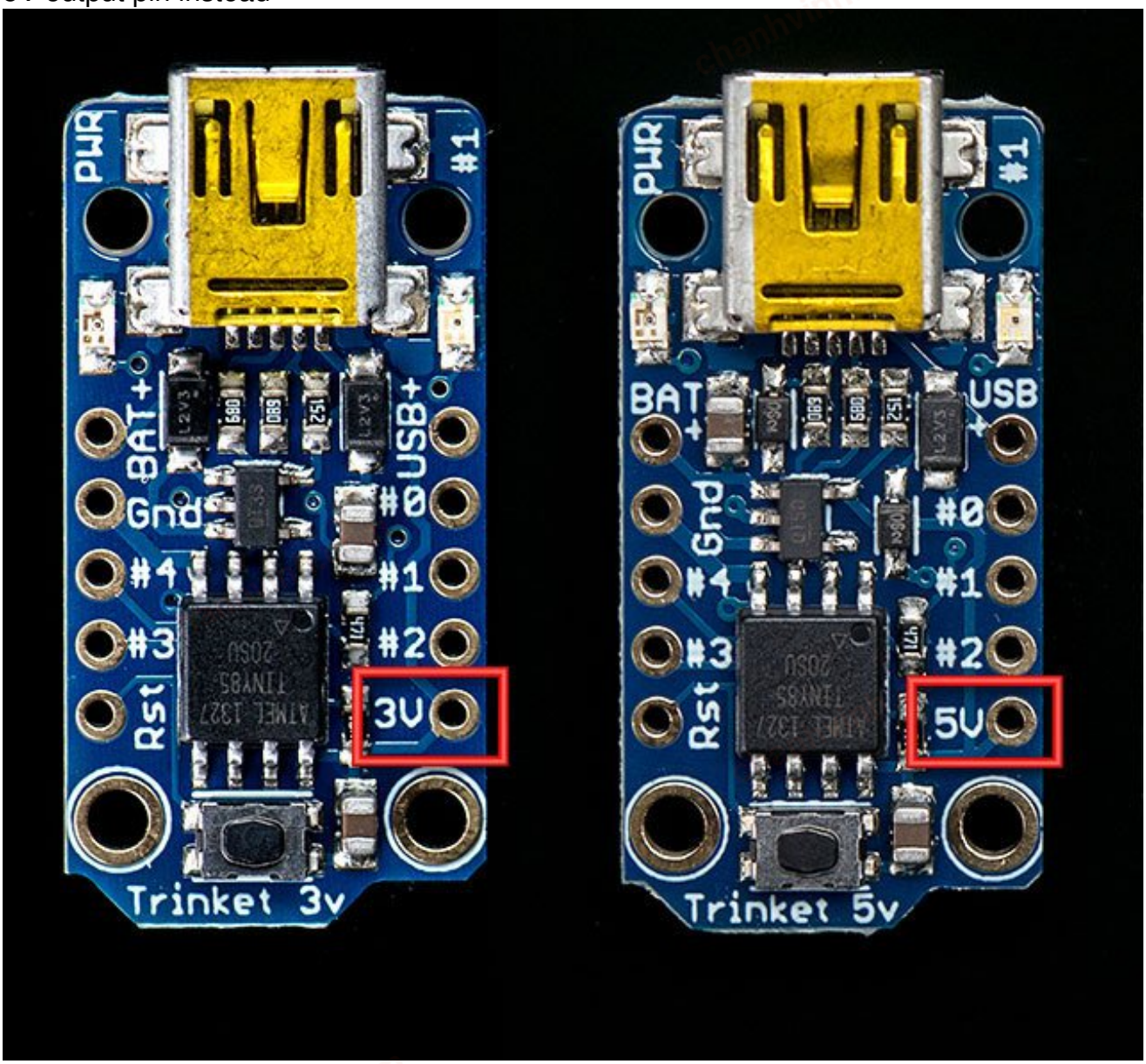
- **Mini-B USB connector** - We went with the tried and true mini-B USB connector for power and/or USB bootloading. In our experience, Micro-B connectors can rip off the PCB easily, but we have not had that problem with mini B, its much more rugged for DIY electronics. It's also a proper USB connector, so you can use any length cable. Some Attiny85 boards use a PCB that slides into a USB port to cut costs, but that makes it hard to re-program and annoying to power with an external battery pack
- **Green Power LED** - you'll know that the board is powered up when this bright LED is lit
- **Red #1 LED** - this LED does double duty. Its connected with a series resistor to the digital #1 GPIO pin. It pulses nicely when the Trinket is in bootloader mode, and its also handy for when you want an indicator LED.
- **Battery + Input** - take your Trinket anywhere and power it from an external battery. This pin can take up 16V DC input, and has reverse-polarity, over-curent and thermal

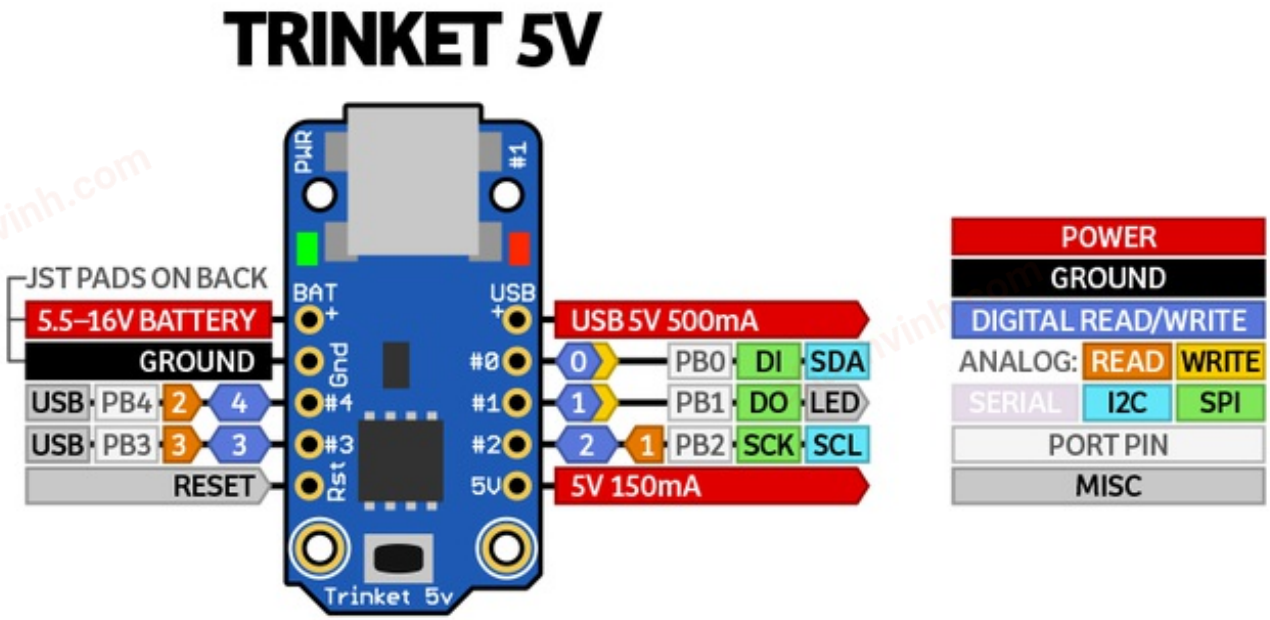
protections. The circuitry inside will use either the battery or USB power, safely switching from one to the other. If both are connected, it will use whichever has the higher voltage

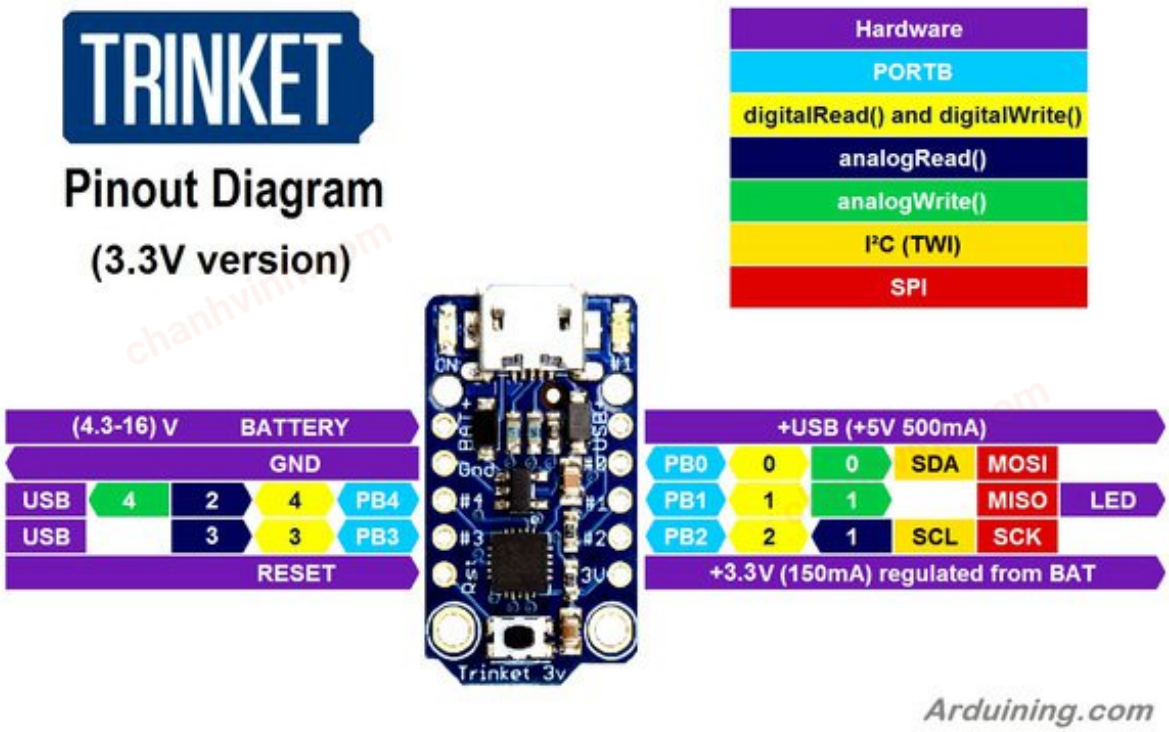
- **USB Power Output** - You can also snag the 5V power from the USB jack in case you need 500mA+ current from your computer or portable USB power pack.
- **Breadboard friendly pins** - You can easily plug this into a little breadboard and have plenty of space for working and plugging stuff in
- **GPIO!** - 5 GPIO pins, at 3 or 5V logic, check the next section for a detailed pinout guide
- **3 or 5V output** - an onboard regulator provides 3.3V or 5V output for powering LEDs, sensors, small motors, etc.
- **Reset Button** - an onboard reset button will launch the bootloader when pressed and the Trinket is plugged into a computer. If it is not connected to a computer, it's smart enough to go straight to the program.
- **External Reset Pin** - we bring out the reset pin so you can reset or restart your Trinket on the road. If the Trinket is in a box or otherwise hard to get to, you can wire up a button to this pin for an external reset button.
- **Four mounting holes** make it easy to attach with 2mm screws or even tiny zip-ties, string, etc.

Pinouts

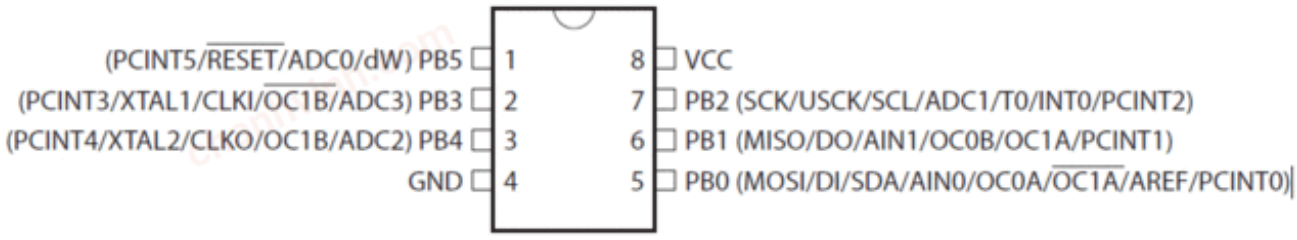
There are two versions of the Trinket: 3V and 5V. They are almost identical but there are slight differences in the pinouts: one has a 3V output pin in the bottom right, the other has a 5V output pin instead







Here's the ATtiny85 pinout



Power Pins

We'll start with the top pins**BAT+** and **USB+** and **GND**

- **BAT+** is the **Battery + Input** pin. If you want to power the trinket from a battery or power adapter or solar panel or any other kind of power source, connect the **+** (positive) pin here! You can connect up to 16V DC. If you have a 3V Trinket, you'll want at least 3.5V input to get a good 3.3V output. If you have a 5V trinket, 5.5V or higher is suggested. This input is reverse-polarity protected.
- **USB+** is the **USB + Output** pin. If you want to use the USB 5V power for something, like charging a battery, or if you need more than 150mA of current (this pin can supply 500mA+ from USB ports) or to detect when the Trinket is plugged into USB, this pin will have 5V power on it if and only if its plugged into something via the mini B connector
- **GND** is the common ground pin, used for logic and power. It is connected to the USB ground and the power regulator, etc. This is the pin you'll want to use for any and all ground connections

GPIO Pins

Next we will cover the 5 GPIO (General Purpose Input Ouput) pins! For reference you may want to also check out the datasheet-reference above for the core ATtiny85 pin

All the GPIO pins can be used as digital inputs, digital outputs, for LEDs, buttons and switches etc. They can provide up to 20mA of current. Don't connect a motor or other high-power component directly to the pins! Instead, use a transistor to power the DC motor on/off (http://adafru.it/aUD)

On a 3V Trinket, the GPIO are 3.3V output level, and should not be used with 5V inputs. On a 5V Trinket, the GPIO are 5V output level, and can be used with 3V inputs but may damage electronic devices that are 3V input only!

The first 3 pins are completely 'free' pins, they are not used by the USB connection so you never have to worry about the USB interface interfering with them when programming

- **GPIO #0** - this is connected to **PB0** on the ATtiny85. This pin can be used as a PWM output, and is also used for I2C data, and SPI data input.
- **GPIO #1** - this is connected to **PB1** on the ATtiny85. This pin can be used as a PWM

output, and is also used for SPI data output. This pin is also connected to the onboard LED (like pin 13 on a regular Arduino).
- **GPIO #2** - this is connected to **PB2** on the ATtiny85. This pin can be used as an analog input (known as **Analog A1**), and is also used for I2C clock and SPI clock.

USB Pins

The next 2 pins are also used for USB programming. That means that when the Trinket is connected to a computer and in bootloader mode or in the middle of uploading a new program, they are used for sending data to/from the computer! It's possible to share these pins if you are careful. The best use of these pins is as **outputs** to things like LEDs. We didn't want to keep these pins off the board but we strongly recommend not using them unless you're sure you need them since you might have to disconnect any connections to reprogram the Trinket!

- **GPIO #3** - this is connected to **PB3** on the Attiny85. This pin is used for USB programming, but its also an analog input known as **Analog A3**
  This pin has a 1.5K pullup to 3.3V built into the Trinket, for USB comm so it may be difficult to use for analog or digital input.
- **GPIO #4** - this is connected to **PB4** on the Attiny85. this pin is used for USB programming, but it can also be used as a PWM analog output and an analog input known as **Analog A2**

Note the numbering of analog pins: Pin 2 is Analog 1, Pin 3 is Analog 3, Pin 4 is Analog 2. For the Uno, the terms A1, A2, and A3 are mapped for you. For ATtiny85's, they are not. So for the pinMode calls, use the Pin number (stenciled on Trinket), for analogRead, use the analog number.

Reset and Regulator Output

The final two pins are at the bottom of the board.

First is the **Rst** reset pin. This is connected directly to the ATtiny85's reset pin and also the reset button which is right next to it. The reset pin is used to enter the bootloader and to reset the board in case you want to restart it. It's also possible to use this pin to re-program in the bootloader or completely remove the bootloader if you have an AVR programmer such as an AVR Dragon, MKii or USBtinyISP. If you want to re-program the board when its in an enclosure or box or otherwise hard to reach, wire a simple button from the RST pin to ground and press it to enter the bootloader for 10 seconds. The #1 LED will pulse to let you

know. The reset button cannot be used as a GPIO, but we think its a lot more useful as a proper reset button!

Lastly we have the regulator output pin. There is an onboard mini power regulator that will take up to 16V DC from the **BAT+** or **USB** connection and regulate it down to a steady 3.3V or 5.0V DC so its safe to use with your sensors and LEDs. On a 3V Trinket, this output will be about 3.3V. On a 5V Trinket, this output will be 5V so be aware in case you want to swap one with the other. You can draw up to 150mA output from this pin. If you need more current, you may want to get it directly from the **USB+** pin, which supplies 5V @ 500mA from a computer or wall adapter

Starting the Bootloader

About the bootloader

A *bootloader* is a tiny piece of software residing on the microcontroller that that helps load your own code into the remaining space.

One of the challenges with the Trinket is that we wanted to have a built-in USB bootloader, but the ATtiny85 doesn't have built-in USB hardware! There are existing USB bootloaders that can work on the 't85 but they use other companies' USB VID/PIDs. Since it not permitted by the USB developer's group (http://adafru.it/cDW) to use others' VID/PIDs we had to adapt one of these existing bootloaders to use our USB ID, but we also wanted to not have to re-compile avrdude or the Arduino IDE since that's such a pain.

So instead, Frank (our awesome engineer with mad USB chops)(http://adafru.it/cDX) created a USB bootloader that combines the elegance of V-USB with the well-supported and tested nature of the USBtinyISP. This bootloader looks just like a USBtinyISP - and since it uses the unique Adafruit VID/PID we own and that we added to avrdude so long ago, it works with only very minimal configuraton tweaks. No need to recompile anything, whew!
Please note: you cannot use the Adafruit USB VID/PID for your own non-Trinket products or projects. Purchase a USB VID for yourself at http://www.usb.org/developers/vendor/

Trinket USB Drivers for Windows

If you're using Windows be sure to follow theWindows Driver Installation page (http://adafru.it/jA9) to install the appropriate drivers for your Trinket.

For Mac OSX or Linux you don't need to install any drivers.

Special Notes on using Trinket with Linux

Trinket is not supported on Linux operating system at this time - try Mac OS or Windows! However, you can try the following - it does work for some computers

Linux is fairly picky about who can poke and prod at the USB port. You can always run **avrdude** or **Arduino IDE** as root, which will make sure you have the proper permissions. If you want to be super-cool you can add a *udev* rule which will let any user (who is not root) connect to the USBtiny driver. That way you don't have to be root all the time!

Check http://learn.adafruit.com/usbtinyisp/avrdude#for-linux (http://adafru.it/cf3) for what to add to your udev file.

How to start the bootloader

Before you try to upload code to the Trinket it must be in the Bootloader Mode. That means its listening for a sketch or program to be sent to it
When the Trinket is in bootloader mode, the red LED will be pulsing. Once the red LED stops pulsing, you must press the reset button to re-enter bootloader mode
The Trinket must be connected to a computer via a USB cable to enter bootloader mode. You can enter the bootloader mode by pressing the little button on the board with your fingernail. The bootloader will 'time out' after 10 seconds, so to re-enter the bootloader mode just re-press the button!

Don't press-and-hold the reset button, be sure to press-and-release!

See the video below for what it looks like to plug it in, have the LED pulse in bootloader mode, time out and then press reset to restart the bootloader

Windows Driver Installation

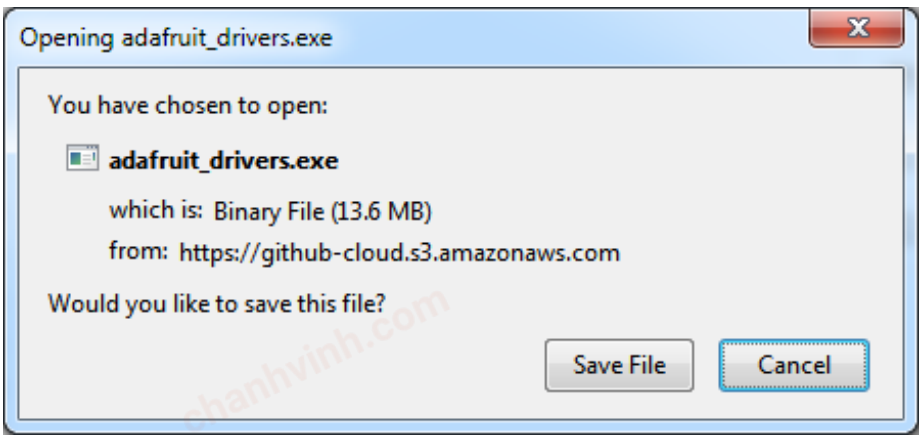
Mac and Linux do not require drivers, only Windows folks need to do this step

Before you plug in your board, you'll need to possibly install a driver!

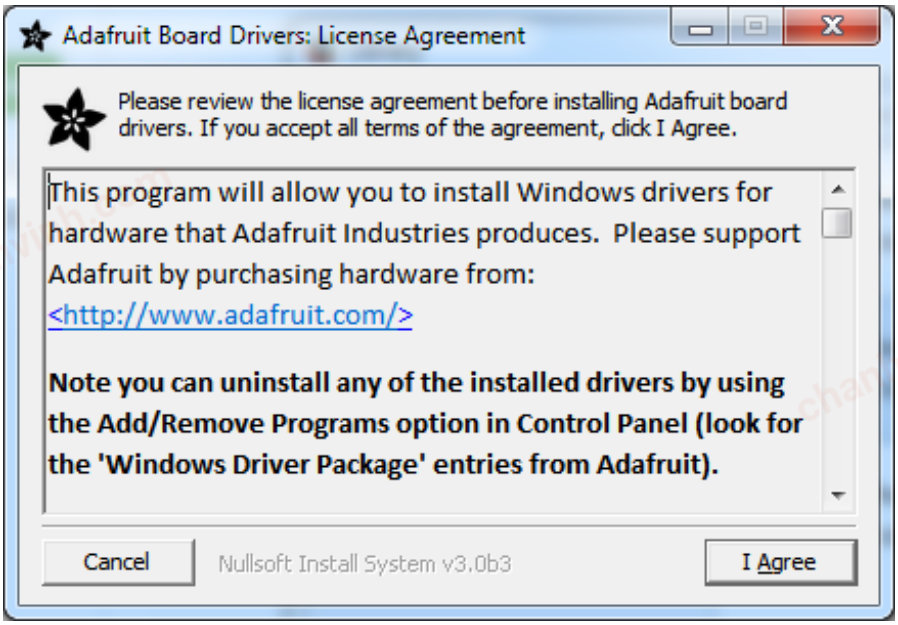
Click below to download our Driver Installer

[Download Adafruit Windows Driver Installer](http://adafru.it/mai)
http://adafru.it/mai

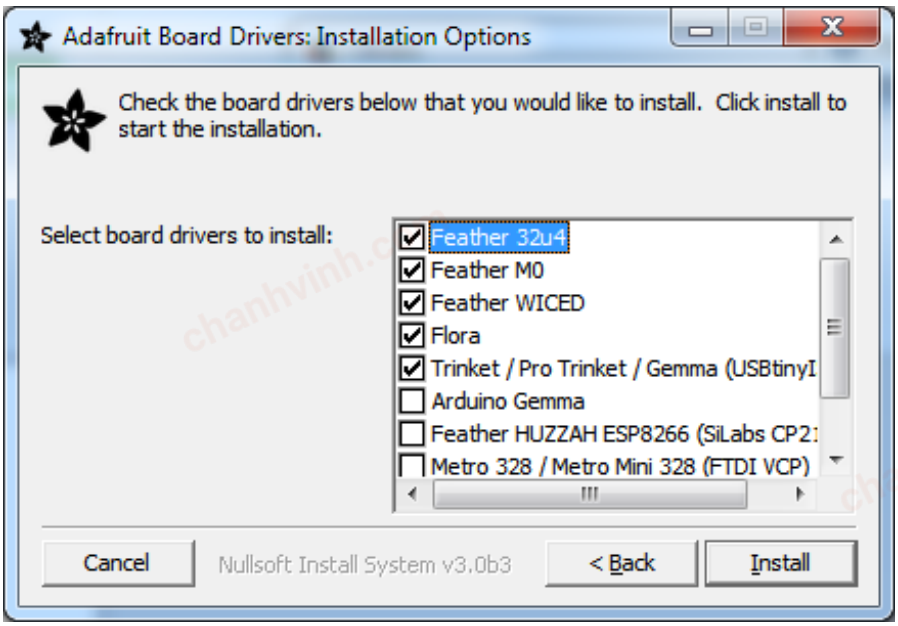
Download and run the installer



Run the installer! Since we bundle the SiLabs and FTDI drivers as well, you'll need to click through the license



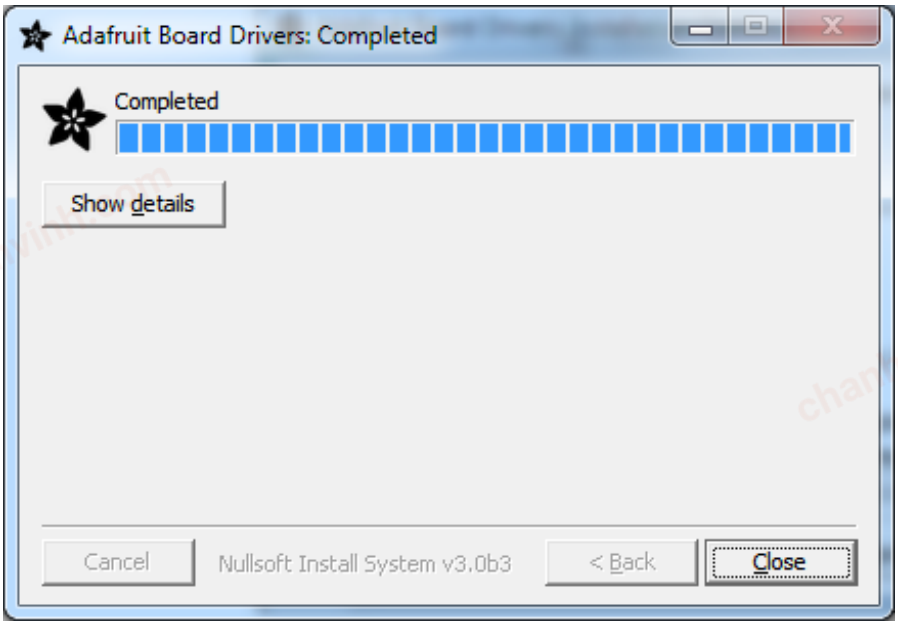
Select which drivers you want to install:



By default, we install the **Feather 32u4** , **Feather M0**, **Flora** and **Trinket** / **Pro Trinket** / **Gemma** / **USBtinyISP** drivers.

You can also, optionally, install the **Arduino Gemma** (different than the Adafruit Gemma!), Huzzah and Metro drivers

Click **Install** to do the installin'


Adafruit Board Drivers: Completed
Completed
Show details
Cancel
Nullsoft Install System v3.0b3
< Back
Close

Setting up with Arduino IDE

Chances are, you picked up a Trinket because it is programmable with the Arduino IDE. Note that the Trinket is not a full Arduino-compatible, it uses a different (smaller) chip than the Uno, Mega, Leonardo or Due. However, there are many small sketches and libraries that will work just fine. Some may not even need anything other than pin number changes. Even though Trinket has a USB connector, it does not have a "Serial Console" capability, so you cannot use Serial to send and receive data to/from a computer!

Arduino IDE Setup

Just follow the steps in the steps in the**Adafruit Arduino IDE setup guide (http://adafru.it/jAc)** to easily add support for Trinket, Gemma, Pro Trinket & more to the Arduino IDE.

You will need Arduino IDE 1.6 or greater

When you're finished installing the IDE & board manager add on come back to this page to continue the Trinket guide.

Blink!

After installing the Arduino IDE with support for Adafruit's boards you can load a simple blinking LED example to test uploading to Trinket works as expected. Open the Arduino IDE and replace the sketch code with the following blink code:

If you are using Linux you may have to be "root" running the Arduino program to have access to the USB port (or add udev exceptions)

```
  /*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.

This example code is in the public domain.

To upload to your Gemma or Trinket:
1) Select the proper board from the Tools->Board Menu
2) Select USBtinyISP from the Tools->Programmer
3) Plug in the Gemma/Trinket, make sure you see the green LED lit
4) For windows, install the USBtiny drivers
```

```
  5) Press the button on the Gemma/Trinket - verify you see
     the red LED pulse. This means it is ready to receive data
  6) Click the upload button above within 10 seconds
*/

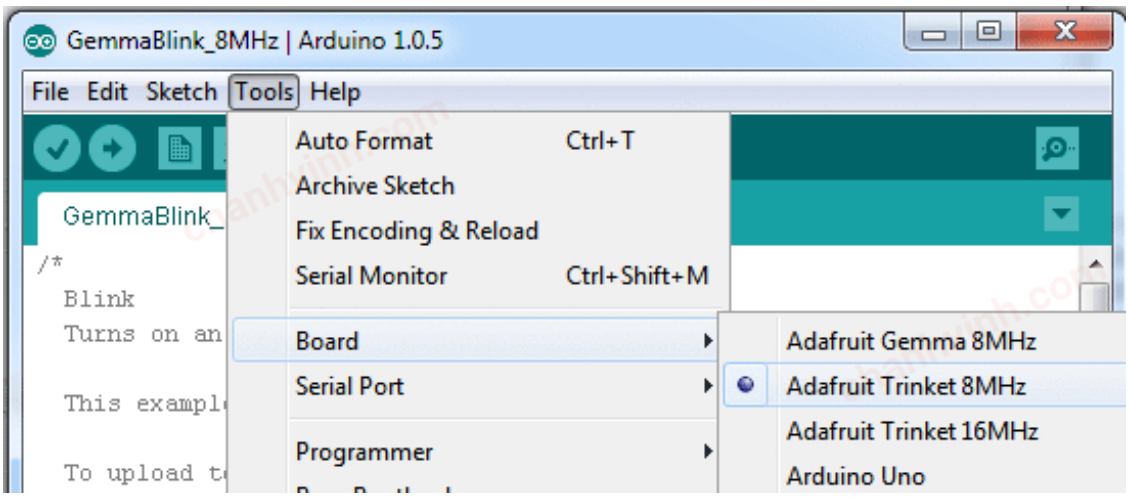
int led = 1; // blink 'digital' pin 1 - AKA the built in red LED

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);

}

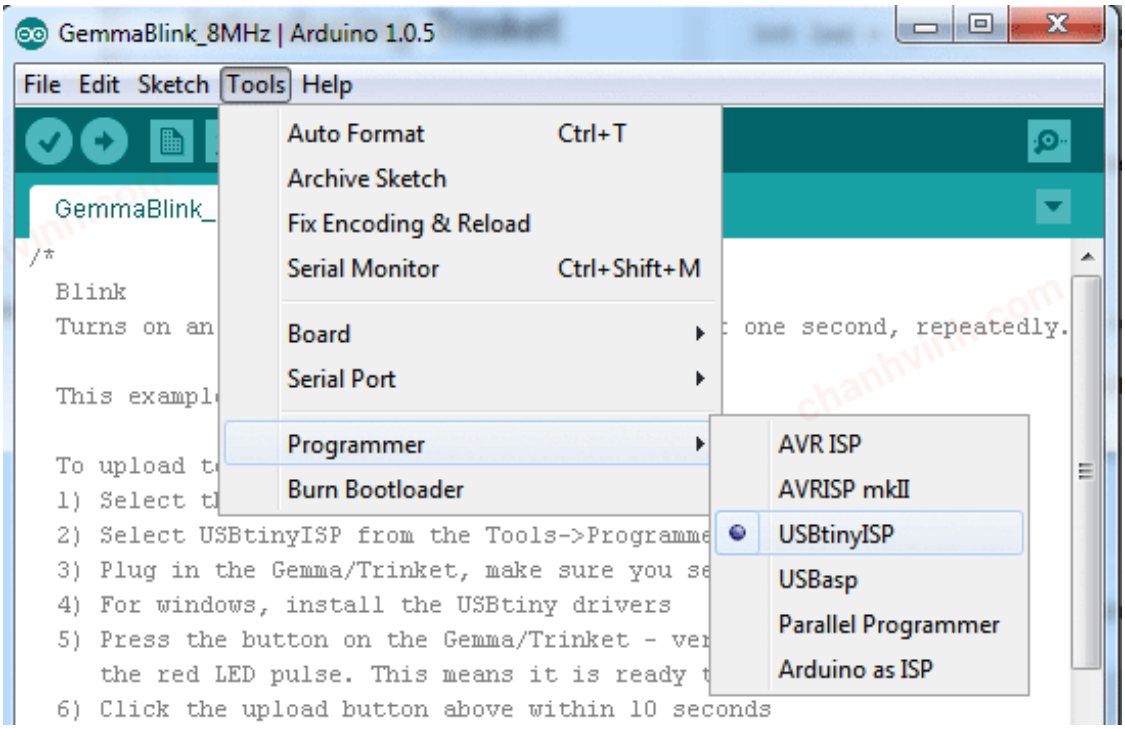
// the loop routine runs over and over again forever:
void loop() {
    digitalWrite(led, HIGH);
    delay(1000);
    digitalWrite(led, LOW);
    delay(1000);
}
```

Select the **Trinket 8MHz** board from the **Tools->Board** menu



Then, select **USBtinyISP** from the **Tools->Programmer** sub-menu

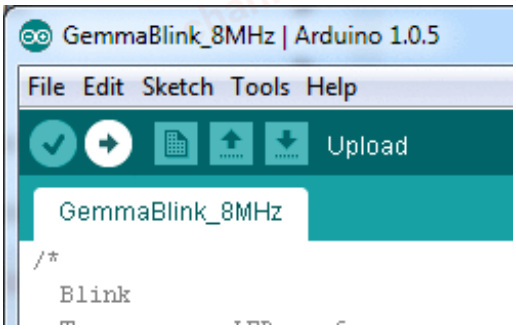
When you're ready to upload, make sure the "Programmer" in the Tools menu is set to USBtinyISP!



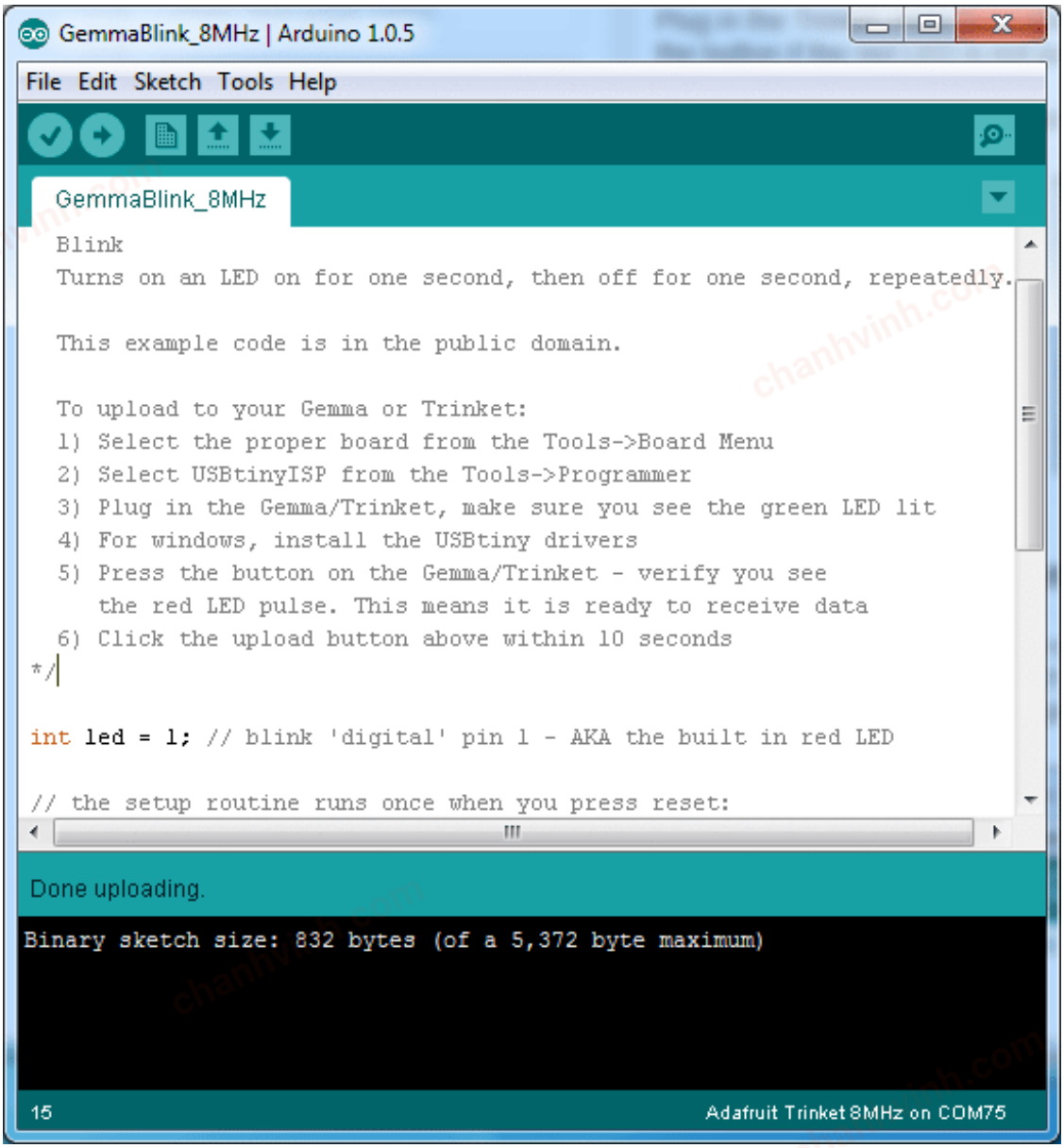
Plug in the Trinket, make sure you see the green LED lit (power good) and the red LED pulsing.

Press the button if the red LED is not pulsing, to get into bootloader mode.

Click the **Upload** button (or select **File->Upload**)



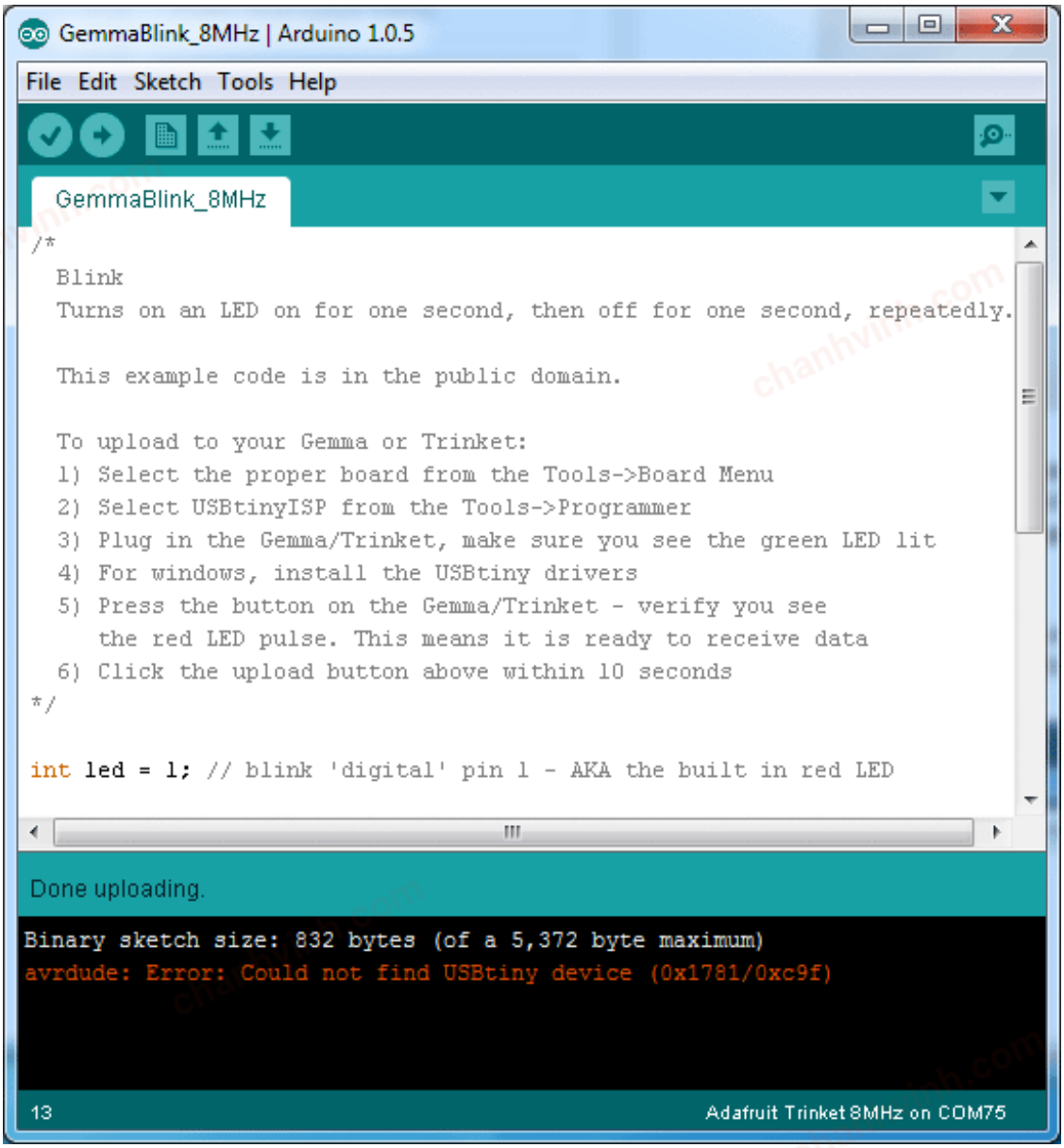
If everything goes smoothly you should see the following (no red error messages) and of course, the red LED on the trinket will blink on/off once a second



Something Went Wrong!

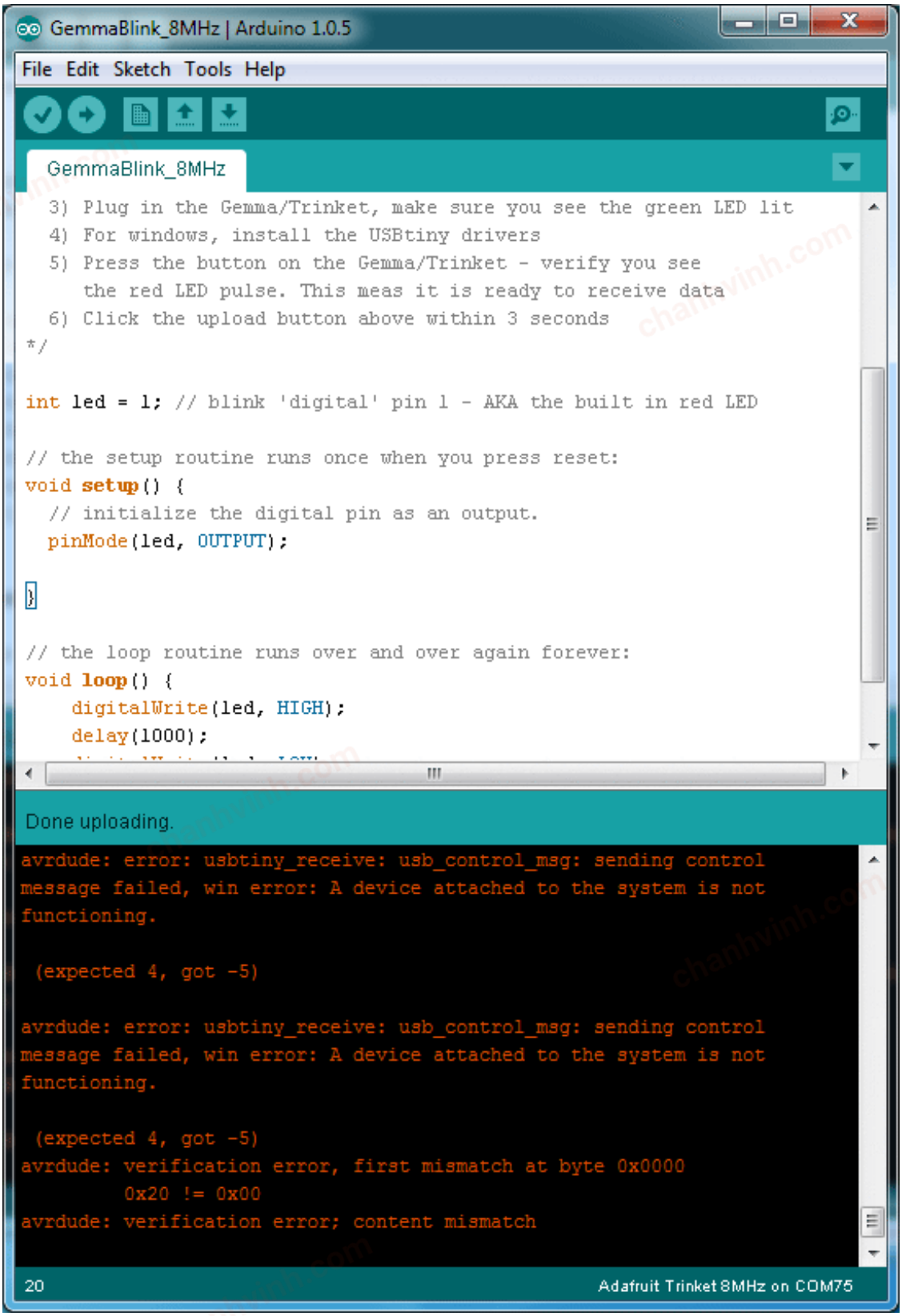
If you get the error message avrdude: Error: Could not find USBtiny device (0x1781/0xc9f)

That means the bootloader wasn't active. Make sure to press the button on the Trinket to activate the bootloader *before* clicking the Upload button.



If you get a lot of red text, errors and also a warning about Verification Failed

Try again! Make sure you have a good USB cable, try connecting through a hub if you have one.



On Linux if you get the error message "usbtiny_receive: error sending control message: Protocol error (expected

4, got -71)"

These can generally be ignored and should not interfere with the program upload.
Unfortunately Linux's USB core is a little flakey communicating with the ATtiny85 processor on the Trinket/Gemma and can cause these errors. If an upload does fail, try it again as it is likely an intermittent issue.

chanhvinh.com

Programming with Arduino IDE

Once you've gotten the basic Blink example to work, you can try some of the other Arduino functions and libraries. We'll be filling out this section with more example code and links to tutorials - this is just to get you started!

pinMode() & digitalWrite() & digitalRead()

You can use pinMode() to make inputs and outputs on any of digital pins #0 thru #4 digitalWrite also works well, and you can also use it with pinMode(INPUT) to activate the internal pull-up resistor on a pin

For example, to set up digital #0 as an input, with an internal pullup, and then check if it is being pulled to ground via a button or switch and turn on the red LED when it is pressed:

```
    /*
  Button
  Turns on an LED when a switch connected from #0 to ground is pressed

  This example code is in the public domain.

  To upload to your Gemma or Trinket:
  1) Select the proper board from the Tools->Board Menu
  2) Select USBtinyISP from the Tools->Programmer
  3) Plug in the Gemma/Trinket, make sure you see the green LED lit
  4) For windows, install the USBtiny drivers
  5) Press the button on the Gemma/Trinket - verify you see
     the red LED pulse. This means it is ready to receive data
  6) Click the upload button above within 10 seconds
*/

#define SWITCH 0
#define LED 1

// the setup routine runs once when you press reset:
void setup() {
  // initialize the LED pin as an output.
  pinMode(LED, OUTPUT);
  // initialize the SWITCH pin as an input.
  pinMode(SWITCH, INPUT);
  // ...with a pullup
  digitalWrite(SWITCH, HIGH);
```

```
}

// the loop routine runs over and over again forever:
void loop() {
  if (! digitalRead(SWITCH)) {  // if the button is pressed
    digitalWrite(LED, HIGH);    // light up the LED
  } else {
    digitalWrite(LED, LOW);     // otherwise, turn it off
  }
}
```

analogRead()

You can read an analog voltage from digital #2 (called **Analog 1**), digital #3 (called **Analog 3**) and digital #4 (called **Analog 2**)

For example, to read an analog voltage on pin #2, you would call **analogRead(1)** to read an analog voltage on pin #4 call **analogRead(2)**

This is a bit confusing because the analog pins are numbered differently than the digital pins!

analogWrite()

There are a few PWM outputs on the Trinket, you can call analogWrite() on digital #0, #1 and #4.

For example, to pulse the built-in LED slowly, upload this code:

```
/*
  Pulse
  Pulses the internal LED to demonstrate the analogWrite function

  This example code is in the public domain.

  To upload to your Gemma or Trinket:
  1) Select the proper board from the Tools->Board Menu
  2) Select USBtinyISP from the Tools->Programmer
  3) Plug in the Gemma/Trinket, make sure you see the green LED lit
  4) For windows, install the USBtiny drivers
  5) Press the button on the Gemma/Trinket - verify you see
     the red LED pulse. This means it is ready to receive data
```

```
  6) Click the upload button above within 10 seconds
*/

int led = 1; // pulse 'digital' pin 1 - AKA the built in red LED

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  for (int i=0; i<256; i++) {
    analogWrite(led, i);  // PWM the LED from 0 to 255 (max)
    delay(5);
  }
  for (int i=255; i>=0; i--) {
    analogWrite(led, i);  // PWM the LED from 255 (max) to 0
    delay(5);
  }
}
```

Make sure you're using the latest Trinket IDE so you can access pin #4's PWM capabilities. If you aren't using the latest IDE you need to manually add functions like the following to init and write analog values to pin #4. However if you have the latest IDE it includes fixes to make pin #4 usable with Arduino's analogWrite function!

```
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%)
}
```

Wire (i2c)

You can use I2C with the Trinket! If you have our board manager package v1.6.4 or later (thats the version of the trinket support package, not IDE) then Wire will work on Attiny85

On the Trinket boards, pin #0 is SDA (I2C data), pin #2 is SCK (I2C clock).

More...

We also know the following libraries work:

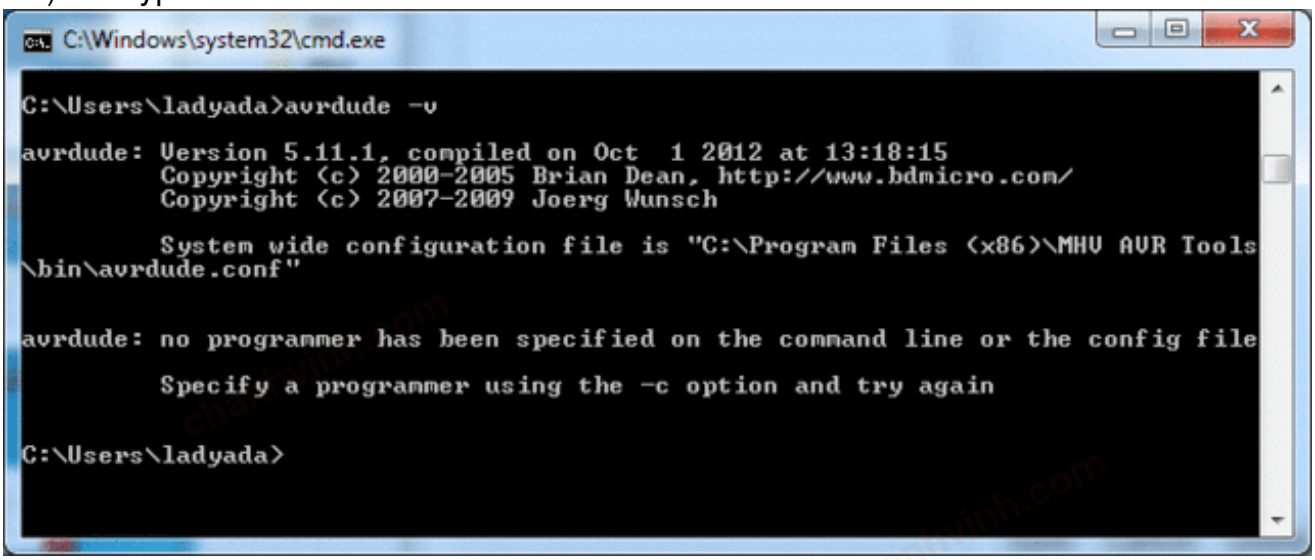
- Adafruit NeoPixel (http://adafru.it/aZU) - control up to ~150 Neopixels via a Trinket!
- SoftwareSerial - the built in SoftSerial library can (at least) transmit data on any digital pin.
- More as we do more testing and verification!

Programming with AVRdude

For more technical users, rather than using the Arduino IDE, they may want to program the Trinket directly with AVR-GCC as the compiler, vi/emacs as their editor and AVRdude as the uploader. That's easy to do!

Target the Attiny85 as the chip used in avr-gcc, with**F_CPU** at 8MHz using the internal oscillator.

To use **avrdude** a minor change must be made to to**avrdude.conf**. To figure out where the **avrdude.conf** is, open up a command window (windows: **cmd**, mac: **Terminal**, linux: **rxvt** etc) and type in **avrdude -v**



Look for the line **System wide configuration file is .....** thats where **avrdude.conf** is.

Because the USB bootloader is a little different than an off-the-shelf programmer, we have to update the configuration file to have a longer erase delay. This does not affect programming bare Attiny85 chips, so you can use this configuration file with Trinkets or raw chips without any problems.

The Short Way

Download the new **avrdude.conf** by clicking on the button, rename the old**avrdude.conf** file to **avrdudeconf.bak** and copy this new one into the same directory

This may not work if you're not running the exact same version of avrdude that we are, in which case, just go down to "the long way" (its really not that long)

[Download the updated Trinket-friendly avrdude.conf](http://adafru.it/cDZ)
http://adafru.it/cDZ
A slightly different configuration file is needed for Mac:
[avrdude.conf (Mac version)](http://adafru.it/cEx)
http://adafru.it/cEx

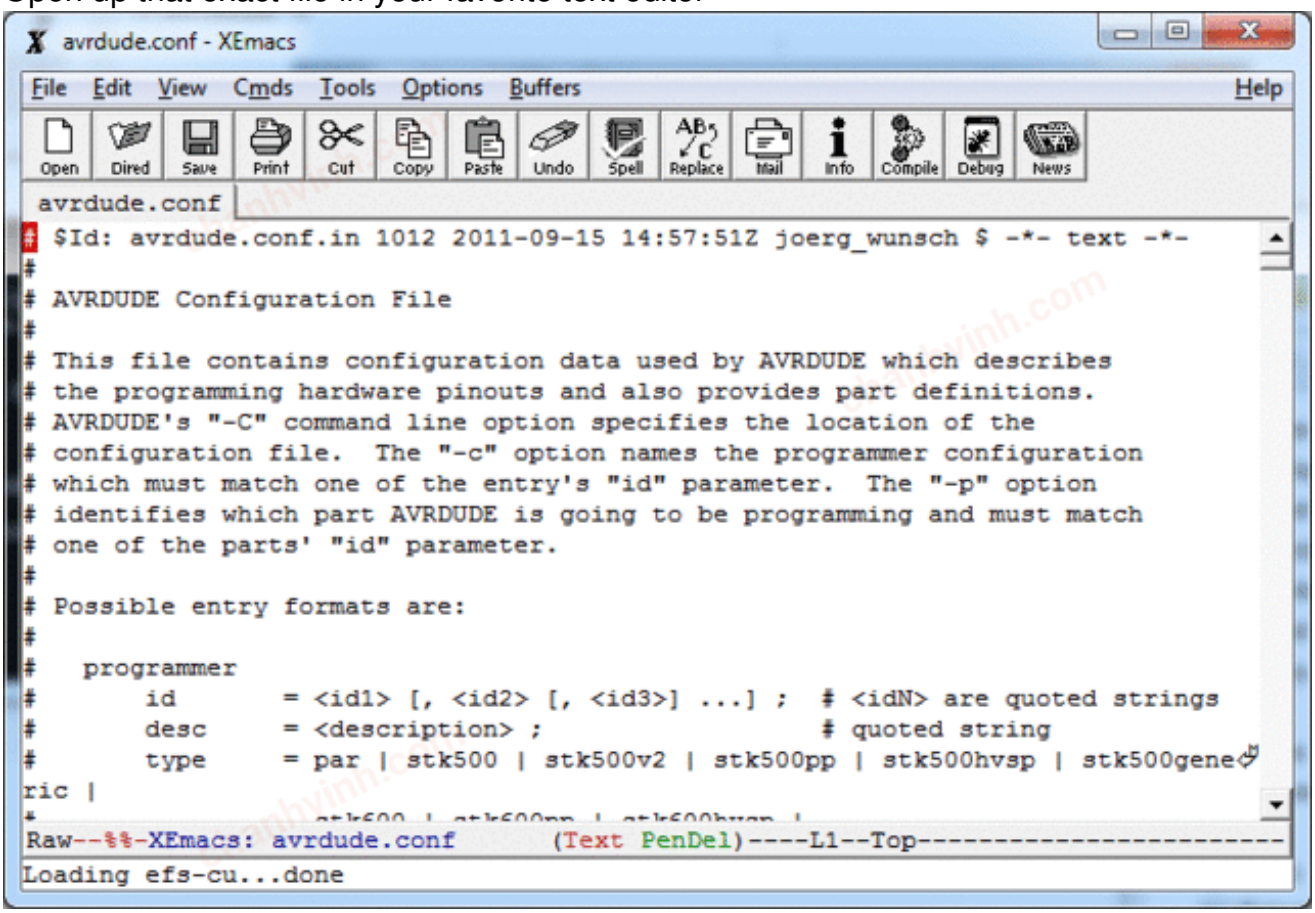
You can also find a Linux version of the avrdude.conf file here:

[avrdude.conf (Linux version)](http://adafru.it/dXe)
http://adafru.it/dXe

The Long Way

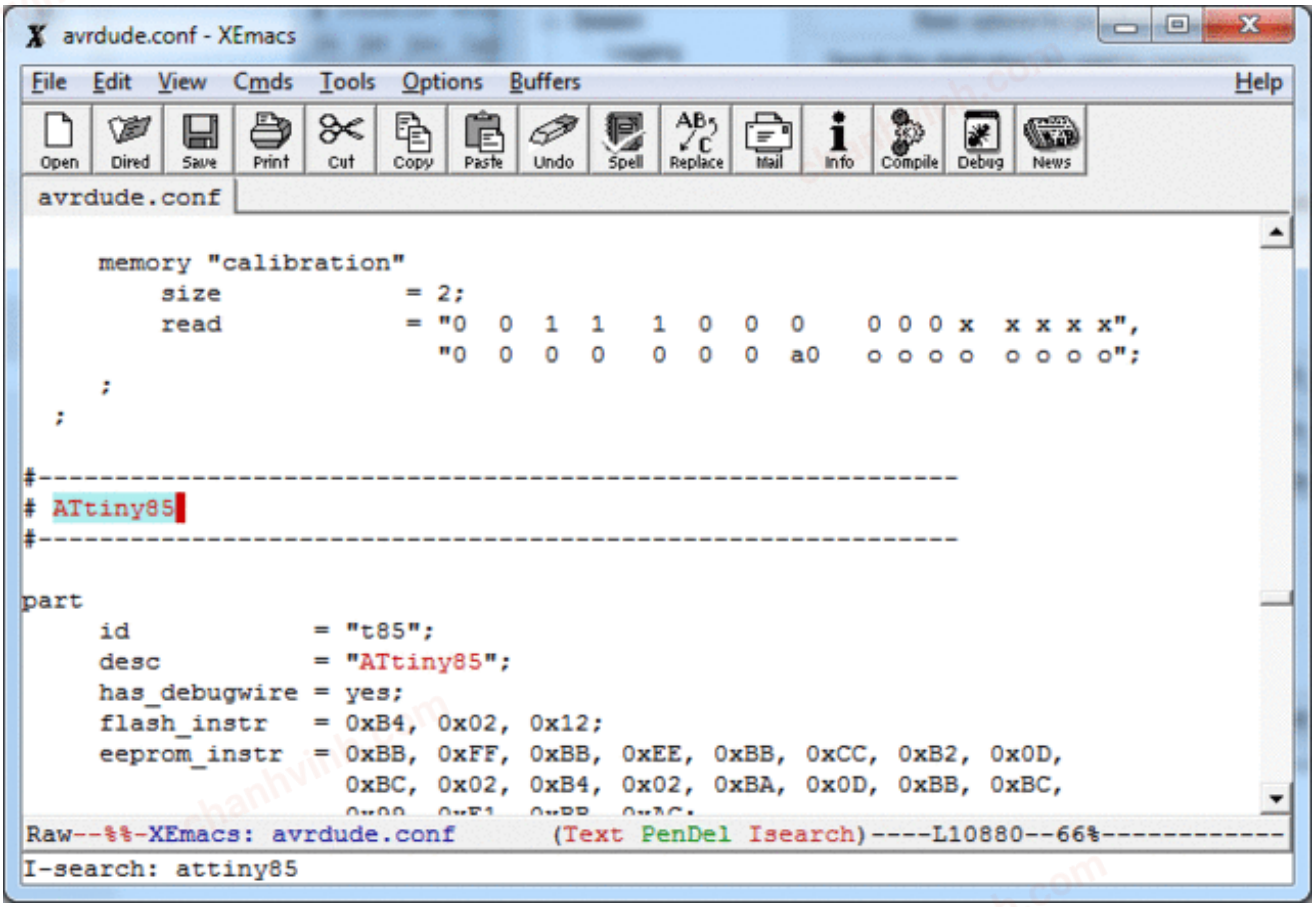
If you want to update your avrdude.conf by hand, its not too hard.

Open up that exact file in your favorite text editor

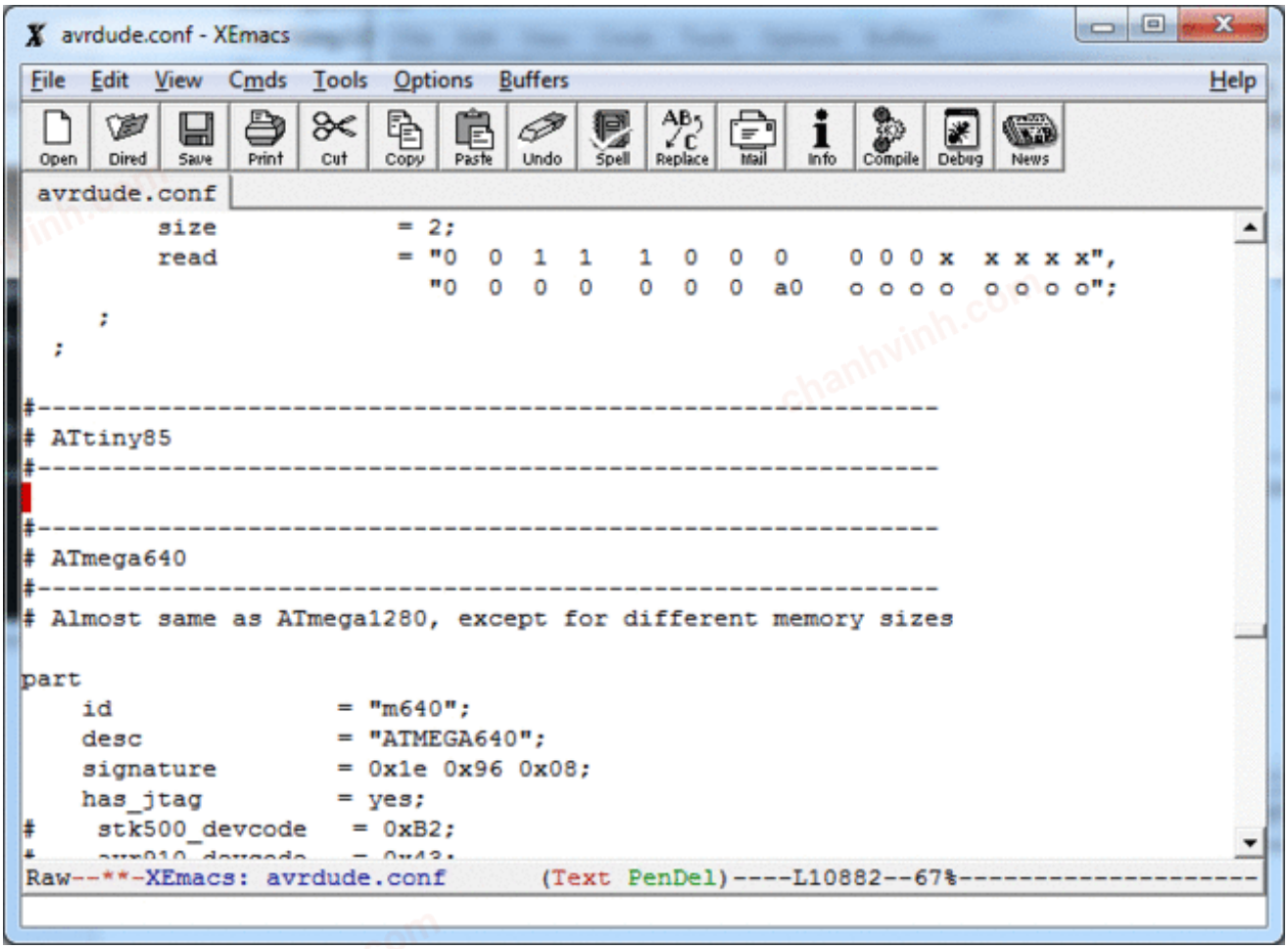


and find the following text

```
#--------------------------------------------------------
# ATtiny85
#--------------------------------------------------------
```



Delete the text after the **Attiny85 header** text starting with **part** and onto until the next header (in ours, that was **ATmega640**



Then paste in the following in the spot where you just deleted!

```
part
    id          = "t85";
    desc        = "ATtiny85";
    has_debugwire = yes;
    flash_instr   = 0xB4, 0x02, 0x12;
    eeprom_instr  = 0xBB, 0xFF, 0xBB, 0xEE, 0xBB, 0xCC, 0xB2, 0x0D,
          0xBC, 0x02, 0xB4, 0x02, 0xBA, 0x0D, 0xBB, 0xBC,
          0x99, 0xE1, 0xBB, 0xAC;
## no STK500 devcode in XML file, use the ATtiny45 one
    stk500_devcode   = 0x14;
##  avr910_devcode   = ?;
##  Try the AT90S2313 devcode:
    avr910_devcode   = 0x20;
    signature       = 0x1e 0x93 0x0b;
    reset           = io;
    chip_erase_delay = 900000;

    pgm_enable      = "1 0 1 0  1 1 0 0   0 1 0 1  0 0 1 1",
                      "x x x x  x x x x   x x x x  x x x x";

    chip_erase      = "1 0 1 0  1 1 0 0   1 0 0 x  x x x x",
                      "x x x x  x x x x   x x x x  x x x x";
```

```
  timeout  = 200;
  stabdelay  = 100;
  cmdexedelay  = 25;
  synchloops  = 32;
  bytedelay  = 0;
  pollindex  = 3;
  pollvalue  = 0x53;
  predelay  = 1;
  postdelay  = 1;
  pollmethod  = 1;

  hvsp_controlstack   =
    0x4C, 0x0C, 0x1C, 0x2C, 0x3C, 0x64, 0x74, 0x66,
    0x68, 0x78, 0x68, 0x68, 0x7A, 0x6A, 0x68, 0x78,
    0x78, 0x7D, 0x6D, 0x0C, 0x80, 0x40, 0x20, 0x10,
    0x11, 0x08, 0x04, 0x02, 0x03, 0x08, 0x04, 0x00;
  hventerstabdelay    = 100;
  hvspcmdexedelay     = 0;
  synchcycles         = 6;
  latchcycles         = 1;
  togglevtg           = 1;
  poweroffdelay       = 25;
  resetdelayms        = 1;
  resetdelayus        = 0;
  hvleavestabdelay    = 100;
  resetdelay          = 25;
  chiperasepolltimeout = 40;
  chiperasetime       = 900000;
  programfusepolltimeout = 25;
  programlockpolltimeout = 25;

   memory "eeprom"
      size            = 512;
     paged            = no;
     page_size        = 4;
      min_write_delay = 4000;
      max_write_delay = 4500;
      readback_p1     = 0xff;
      readback_p2     = 0xff;
      read            = "1  0  1  0   0  0  0  0    0 0 0 x  x x x a8",
                        "a7 a6 a5 a4  a3 a2 a1 a0   o o o o  o o o o";

      write           = "1  1  0  0   0  0  0  0    0 0 0 x  x x x a8",
                        "a7 a6 a5 a4  a3 a2 a1 a0   i i i i  i i i i";

loadpage_lo = "  1   1   0   0     0   0   0   1",
    "  0   0   0   0     0   0   0   0",
    "  0   0   0   0     0   0  a1  a0",
    "  i   i   i   i     i   i   i   i";

writepage = "  1   1   0   0     0   0   1   0",
```

```
    " 0  0  x  x    x  x  x  a8",
    " a7  a6  a5  a4    a3  a2  0  0",
    " x  x  x  x    x  x  x  x";

  mode  = 0x41;
  delay  = 12;
  blocksize = 4;
  readsize = 256;
      ;
    memory "flash"
      paged          = yes;
      size           = 8192;
      page_size      = 64;
      num_pages      = 128;
      min_write_delay = 30000;
      max_write_delay = 30000;
      readback_p1    = 0xff;
      readback_p2    = 0xff;
      read_lo        = " 0  0  1  0  0  0  0  0",
                       " 0  0  0  0 a11 a10  a9  a8",
                       " a7  a6  a5  a4  a3  a2  a1  a0",
                       " o  o  o  o  o  o  o  o";

      read_hi        = " 0  0  1  0  1  0  0  0",
                       " 0  0  0  0 a11 a10  a9  a8",
                       " a7  a6  a5  a4  a3  a2  a1  a0",
                       " o  o  o  o  o  o  o  o";

      loadpage_lo    = " 0  1  0  0  0  0  0  0",
                       " 0  0  0  x  x  x  x  x",
                       " x  x  x  a4  a3  a2  a1  a0",
                       " i  i  i  i  i  i  i  i";

      loadpage_hi    = " 0  1  0  0  1  0  0  0",
                       " 0  0  0  x  x  x  x  x",
                       " x  x  x  a4  a3  a2  a1  a0",
                       " i  i  i  i  i  i  i  i";

      writepage      = " 0  1  0  0  1  1  0  0",
                       " 0  0  0  0 a11 a10 a9 a8",
                       " a7 a6 a5  x  x  x  x  x",
                       " x  x  x  x  x  x  x  x";

  mode  = 0x41;
  delay  = 6;
  blocksize = 32;
  readsize = 256;
      ;
#   ATtiny85 has Signature Bytes: 0x1E 0x93 0x08.
    memory "signature"
      size           = 3;
      read           = "0  0  1  1  0  0  0  0  0  0  0  x  x  x  x  x",
```

```
                "x x x x  x x a1 a0  o o o o  o o o o";
  ;
 memory "lock"
    size          = 1;
    write         = "1 0 1 0  1 1 0 0  1 1 1 x  x x x x",
                    "x x x x  x x x x  1 1 i i  i i i i";
   min_write_delay = 9000;
   max_write_delay = 9000;
  ;

 memory "lfuse"
    size          = 1;
    write         = "1 0 1 0  1 1 0 0  1 0 1 0  0 0 0 0",
                    "x x x x  x x x x  i i i i  i i i i";

    read          = "0 1 0 1  0 0 0 0  0 0 0 0  0 0 0 0",
                    "x x x x  x x x x  o o o o  o o o o";
   min_write_delay = 9000;
   max_write_delay = 9000;
  ;

 memory "hfuse"
    size          = 1;
    write         = "1 0 1 0  1 1 0 0  1 0 1 0  1 0 0 0",
                    "x x x x  x x x x  i i i i  i i i i";

    read          = "0 1 0 1  1 0 0 0  0 0 0 0  1 0 0 0",
                    "x x x x  x x x x  o o o o  o o o o";
   min_write_delay = 9000;
   max_write_delay = 9000;
  ;

 memory "efuse"
    size          = 1;
    write         = "1 0 1 0  1 1 0 0  1 0 1 0  0 1 0 0",
                    "x x x x  x x x x  x x x x  x x x i";

    read          = "0 1 0 1  0 0 0 0  0 0 0 0  1 0 0 0",
                    "x x x x  x x x x  o o o o  o o o o";
   min_write_delay = 9000;
   max_write_delay = 9000;
 ;

 memory "calibration"
    size          = 2;
    read          = "0  0  1  1  1  0  0  0   0 0 0 x  x x x x",
                    "0  0  0  0  0  0  0  0  a0  o o o o  o o o o";
 ;
;
```

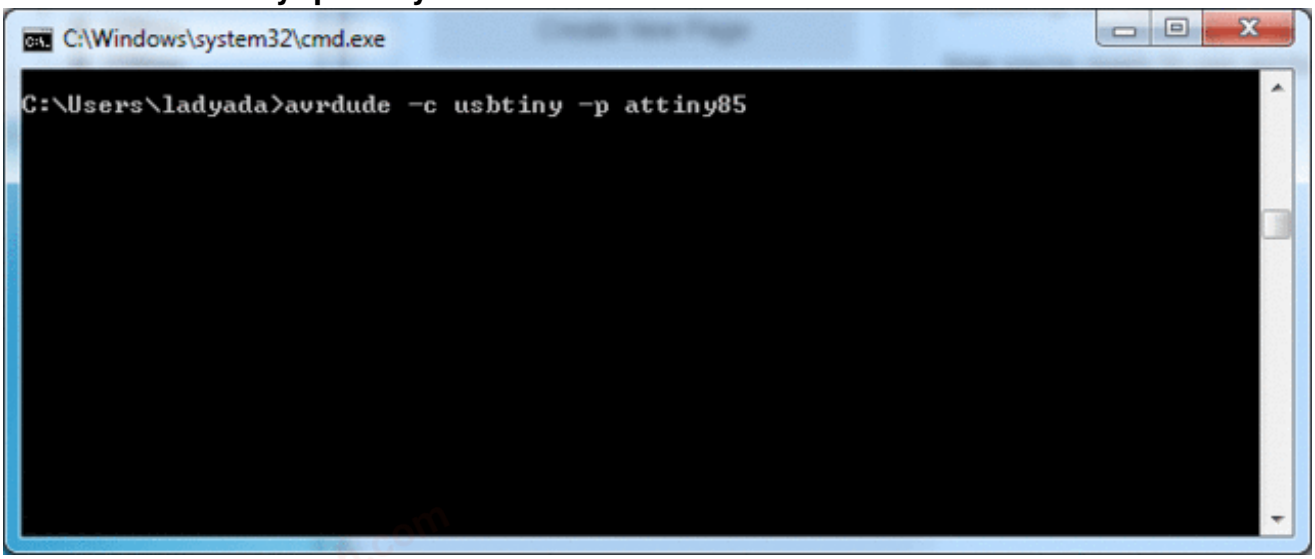
If editing manually, Mac users should delete all references to parallel port programmers

("type = par;").

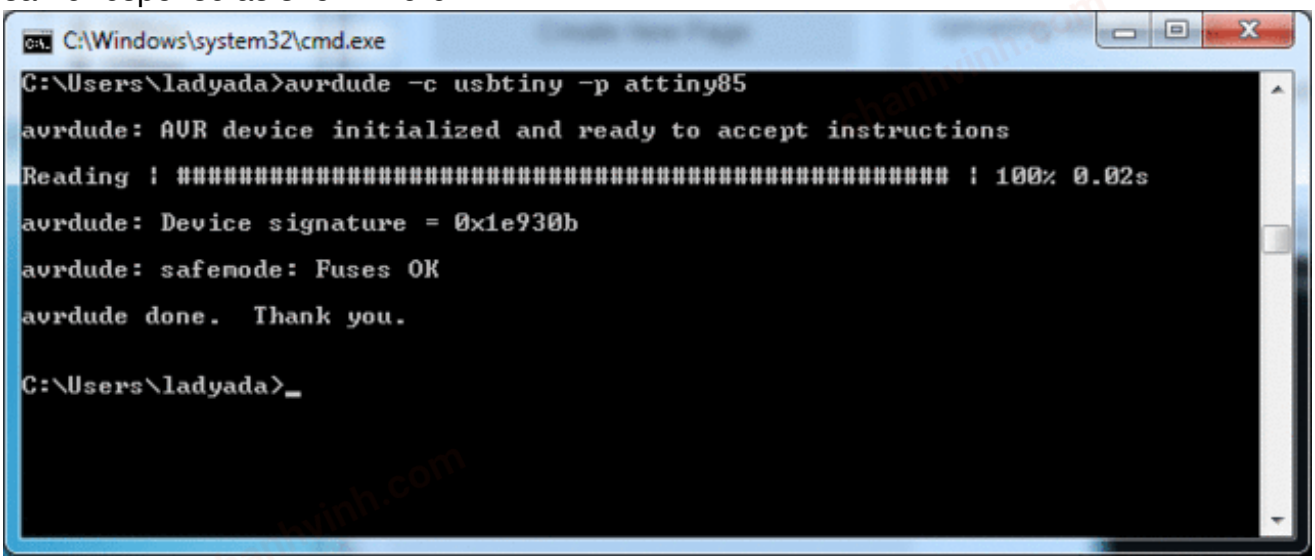
Uploading with AVRdude

Now you're ready to use avrdude. Open up your command line and enter in this line (but don't hit return)

avrdude -c usbtiny -p attiny85



Now plug in the Trinket into the computer's USB port and/or press the reset button to enter the bootloader. You should see the red LED pulsing. Now press return, you should get the same response as shown here:



If you get the response

avrdude: Error: Could not find USBtiny device (0x1781/0xc9f)

the bootloader is not active, make sure you see the red LED pulsing, press the reset button to start the bootloader again.

Programming in a Blink example

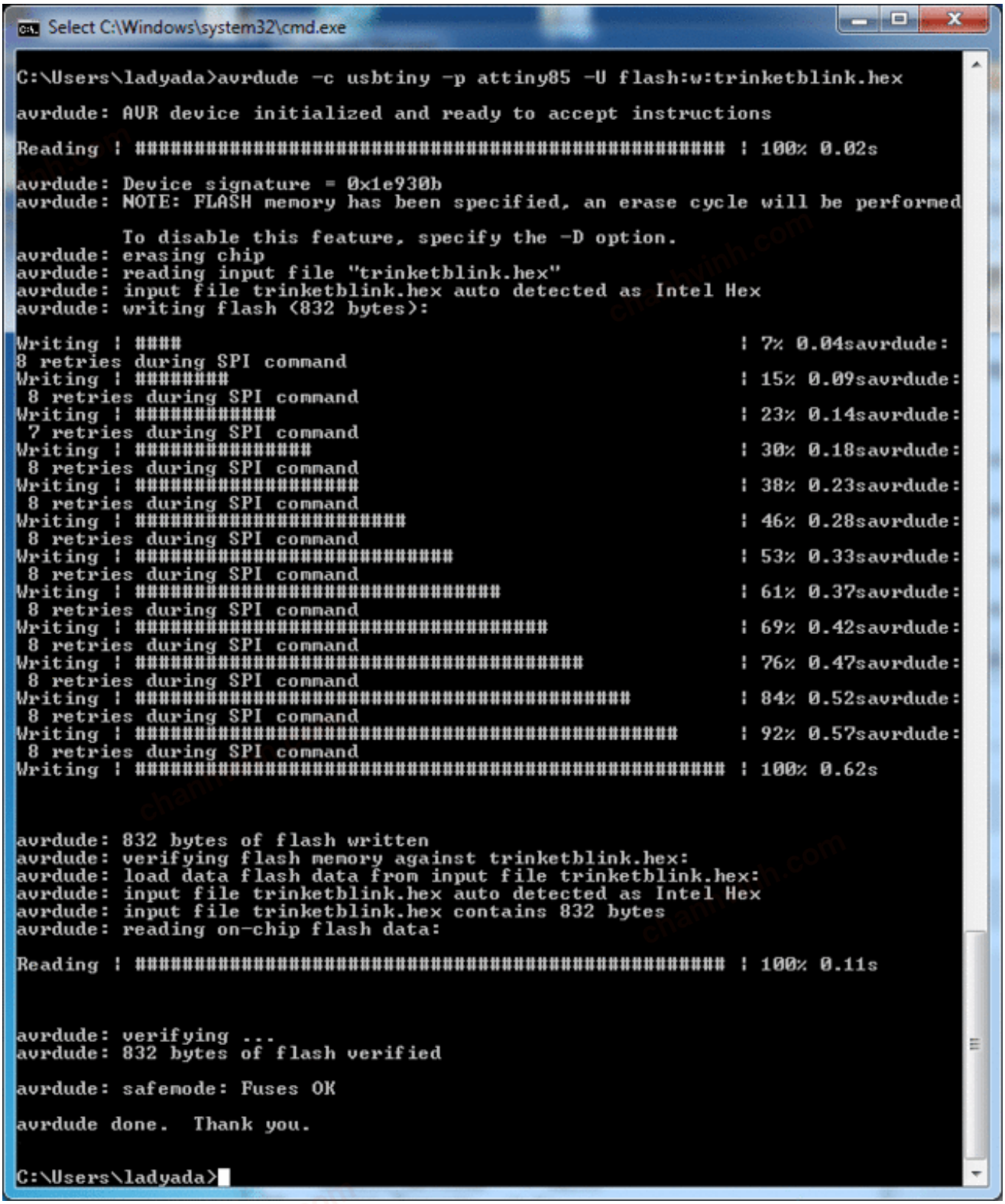
For more details on using avdude and avr-gcc, you'll need to read a detailed tutorial or book on those subjects. However, you can do a basic test by uploading the following HEX file, which will blink the #1 LED once a second on and off. (Its a bit chunky as blink.hex's go as it has all the Arduino IDE stuff in there too. If you wrote it in straight-C it would be like 100 bytes) - if you want to see the source code for this, it's basically the Arduino **file->examples->basics->blink** demo with 1 as the LED pin

Click the button to download it and place it in the same directory as your command prompt, in these screenshots that's **C:\Users\ladyada**

trinketblink.hex
http://adafru.it/cF1
And uploading it with the command **avrdude -c usbtiny -p attiny85 -U flash:w:trinketblink.hex**
or, if that's giving errors, **avrdude -c usbtiny -p attiny85 -D -U flash:w:trinketblink.hex** (note the extra **-D**)

As before, type out the command, then press the reset button to start the bootloader and once the red LED is pulsing, hit return

When uploading, you will see a lot of **avrdude: 8 retries during SPI command** and similar warnings. **THIS IS OK!** Because of the way the ATtiny85 works, there's a small delay when writing the new program to flash, and during that delay, it cannot save the data and also send USB data at the same time. This causes the USB reply to avrdude to be delayed and avrdude to spit out the retry alert.


Select C:\Windows\system32\cmd.exe
C:\Users\ladyada>avrdude -c usbtiny -p attiny85 -U flash:w:trinketblink.hex
avrdude: AVR device initialized and ready to accept instructions
Reading | ## | 100% 0.02s
avrdude: Device signature = 0x1e930b
avrdude: NOTE: FLASH memory has been specified, an erase cycle will be performed
To disable this feature, specify the -D option.
avrdude: erasing chip
avrdude: reading input file "trinketblink.hex"
avrdude: input file trinketblink.hex auto detected as Intel Hex
avrdude: writing flash (832 bytes):
Writing | ####
8 retries during SPI command
Writing | ########
8 retries during SPI command
Writing | ############
7 retries during SPI command
Writing | ################
8 retries during SPI command
Writing | ####################
8 retries during SPI command
Writing | ########################
8 retries during SPI command
Writing | ############################
8 retries during SPI command
Writing | ################################
8 retries during SPI command
Writing | ####################################
8 retries during SPI command
Writing | ##
8 retries during SPI command
Writing | ##
8 retries during SPI command
Writing | ##
8 retries during SPI command
Writing | ## | 100% 0.62s
avrdude: 832 bytes of flash written
avrdude: verifying flash memory against trinketblink.hex:
avrdude: load data flash data from input file trinketblink.hex:
avrdude: input file trinketblink.hex auto detected as Intel Hex
avrdude: input file trinketblink.hex contains 832 bytes
avrdude: reading on-chip flash data:
Reading | ## | 100% 0.11s
avrdude: verifying ...
avrdude: 832 bytes of flash verified
avrdude: safemode: Fuses OK
avrdude done. Thank you.
C:\Users\ladyada>

16MHz vs 8MHz Clock

The Trinket by default runs at 8 MHz, a plenty fast speed for nearly all of your microcontroller needs. However, you may want to use code that requires it to run at 16 MHz, or maybe you just want a little boost in speed.

The 16 MHz clock speed is for Trinket 5V only!

The ATtiny85 is only specified to run at 16 MHz when powered at 5V - that means that *officially* you can only run the 5V Trinket at 16 MHz.
However, the AVR series is pretty forgiving for overclocking, so *you may be able to run the 3V Trinket at 16 MHz*. Note that this is still overclocking, your code may run flakey or not at all! Overclocking should not damage the AVR, but we still recommend sticking with 8 MHz only for the 3V version, and 8 or 16MHz only on the 5V version.

Power Tradeoffs

Doubling the speed will increase the power usage just a bit. At 8 MHz the current draw is about 9 milliamps. That number includes the green power LED which draws about 3mA so thats 6mA for the microcontroller itself.

At 16 MHz the draw is 12mA total. Subtracting the green LED current draw, that means 9mA for the microcontroller itself.

How to activate the 16 MHz clock

...on AVR-GCC

We can activate the 16MHz clock 'in software' simply by asking the chip to set the clock prescale. If you are using raw avr-gcc, run this as the first line in main()

```
clock_prescale_set(clock_div_1);
```

You may need to add #include to your file so that the commands are recognized. Then make sure to compile your code with F_CPU = 16000000

...Arduino IDE

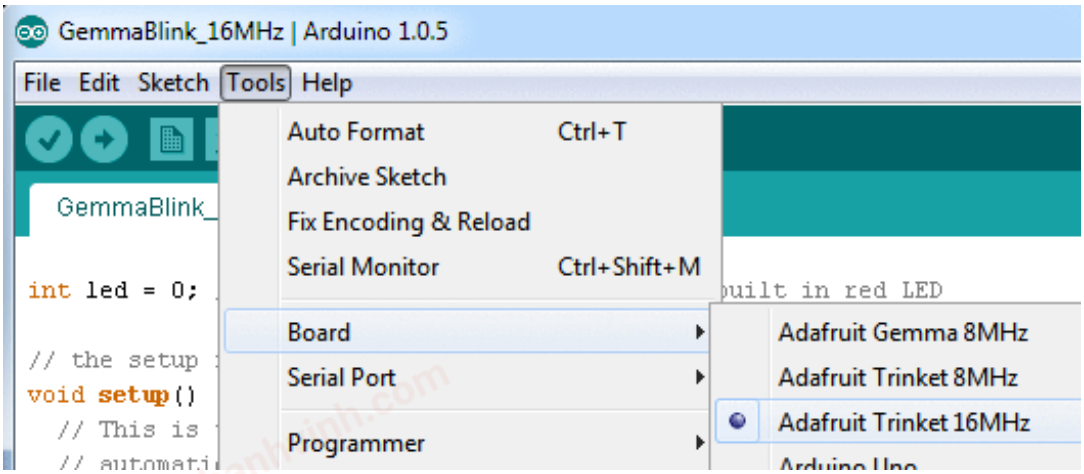
Using 16 MHz mode is very similar when using the Arduino IDE. Add the following line to the very top of your Arduino sketch (as the first line)

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

Then, in **setup()** - add this as the first line in the function:

```
if (F_CPU == 16000000) clock_prescale_set(clock_div_1);
```

Then select Trinket 16MHz from the Tools->Board menu. Your code will compile and run at 16 MHz!



Repairing bootloader

The ATtiny85 does not have a protected-bootloader section. This means its possible to accidentally overwrite the bootloader (or even if you unplug the Trinket while uploading it might have difficulties from then on)

You can use an Arduino UNO to re-program the bootloader onto your Trinket (or Gemma). This loader isn't tested to work with any other kind of Arduino.

Connect:

- Trinket **VBAT+** pin to Arduino 5V (or just power it via a battery or USB cable)
- Trinket **GND** pin to Arduino GND
- Trinket **RST** to Arduino #10
- Trinket **#0** pin to Arduino #11
- Trinket **#1** pin to Arduino #12
- Trinket **#2** pin to Arduino #13

On a Gemma, alligator clips work well. the Reset pin is*underneath* the MiniUSB Jack. You may have to solder a wire temporarily. Alternatively, *sometimes* you can just hold the reset button down while running the sketch (type 'G' to start) and it might work. Soldering a wire works best.

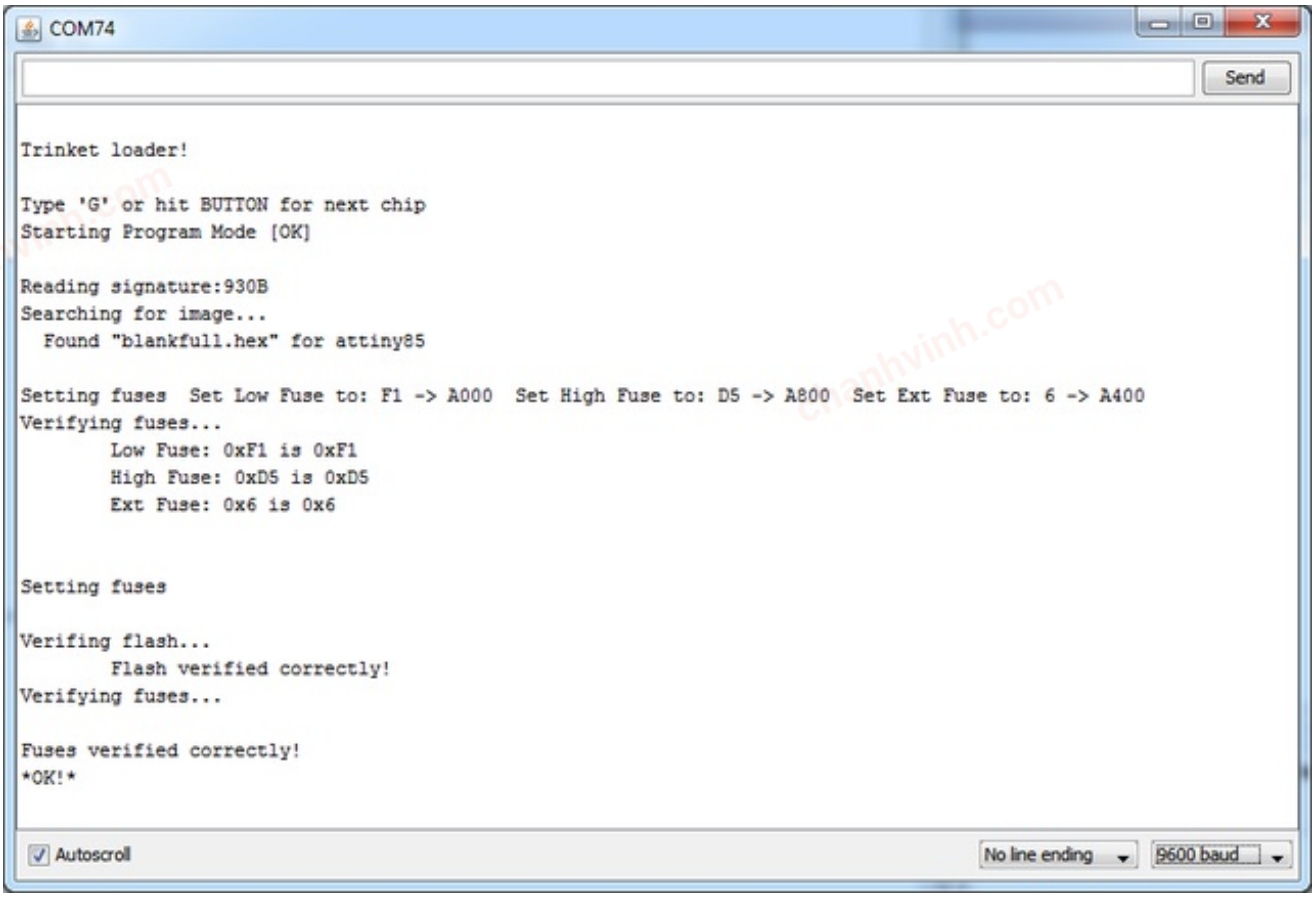
Then download, uncompress and run the Trinketloader sketch, pick the one that matches your Arduino version!

trinketloader_2015-06-09.zip
http://adafru.it/fmg

Uncompress and open in the Arduino IDE, select the UNO and serial port to the UNO Arduino board you are uploading too and upload it to the UNO.

Open up the serial console at 9600 baud and when it tells you to, press the miniature button on the Trinket (or Gemma) or type in G into the serial console and click Send

You should see the following, the fuses, firmware burned and verified! It will take 2 seconds



If you keep getting "Failed to verify fuses" while trying to restore the bootloader on a 3.3V Trinket using a Duemilanove. Try connecting 5V to the 3V pin on the Trinket, not VBAT+. Make sure USB is not plugged into the Trinket when you do this. Its a bit of a hack but may help!

Downloads

Datasheets

- Datasheet for the onboard regulator used (MIC5225 3.3V and 5.0V) (http://adafru.it/dQO)
- Webpage for the ATtiny85, the microcontroller used in the Trinket(http://adafru.it/cE5)

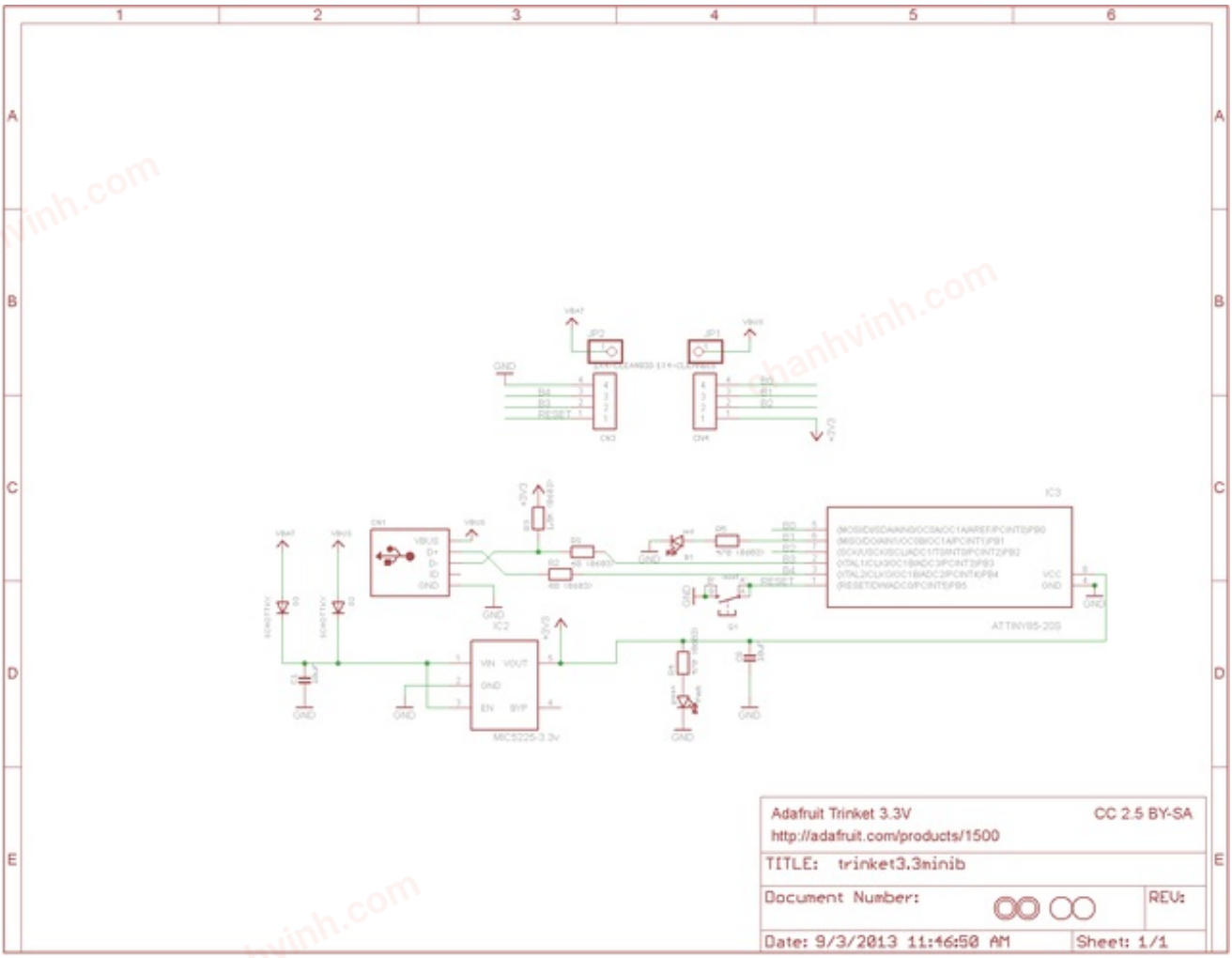
Source code

- Original code for the Trinket/Gemma bootloader on github - (http://adafru.it/cE6)We do not offer any support for this code, it is released as-is!
- PCB files for both Trinkets (http://adafru.it/o7A)- you can use this to get more dimensions or part details, they are the fabrication files
- Fritzing object in the Adafruit Fritzing Library(http://adafru.it/aP3)

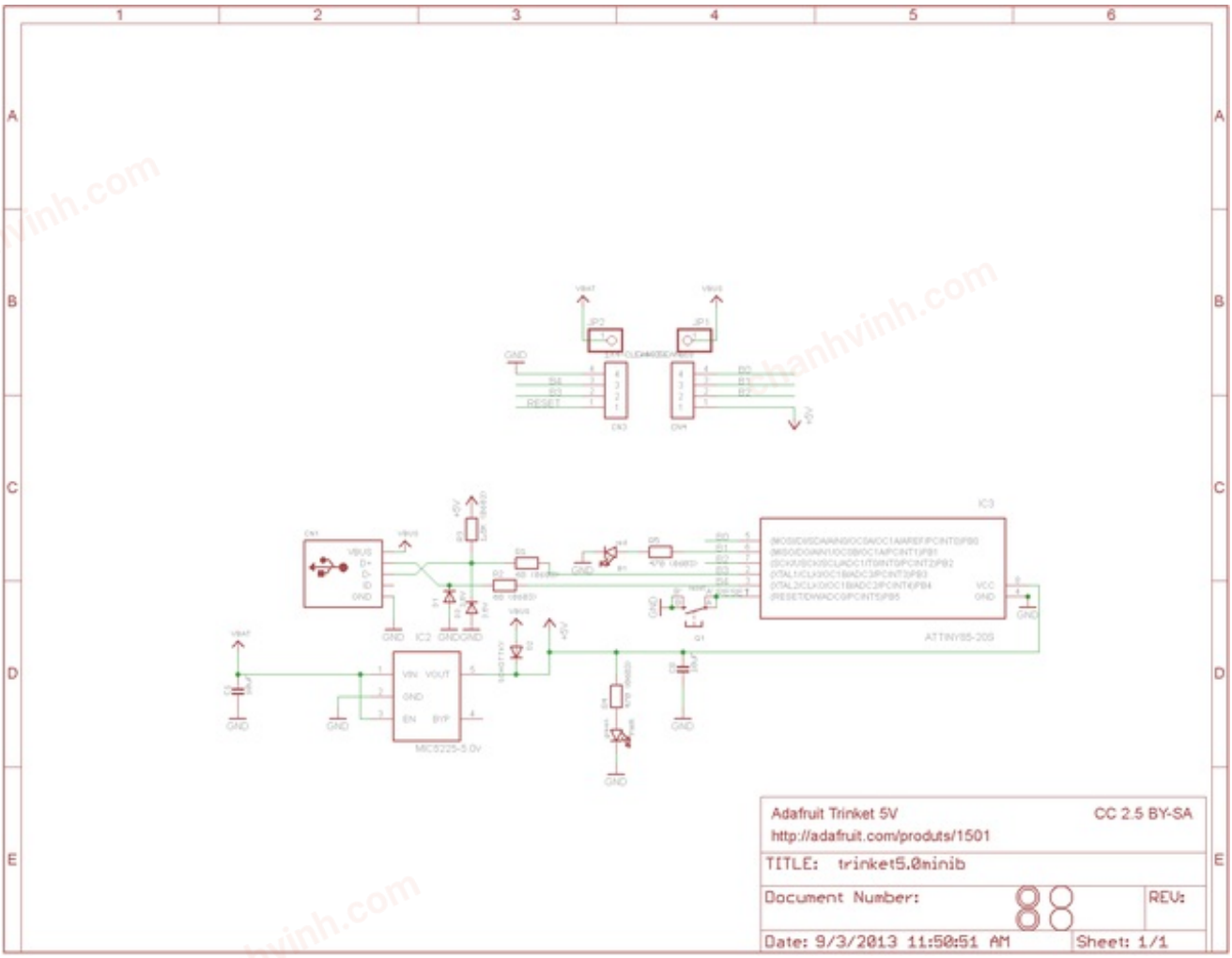
Please note: you cannot use the Adafruit USB VID/PID for your own non-Trinket/Gemma products or projects. Purchase a USB VID for yourself at http://www.usb.org/developers/vendor/

Schematics

Click to embiggen
Trinket 3.3V Schematic:



Trinket 5V schematic

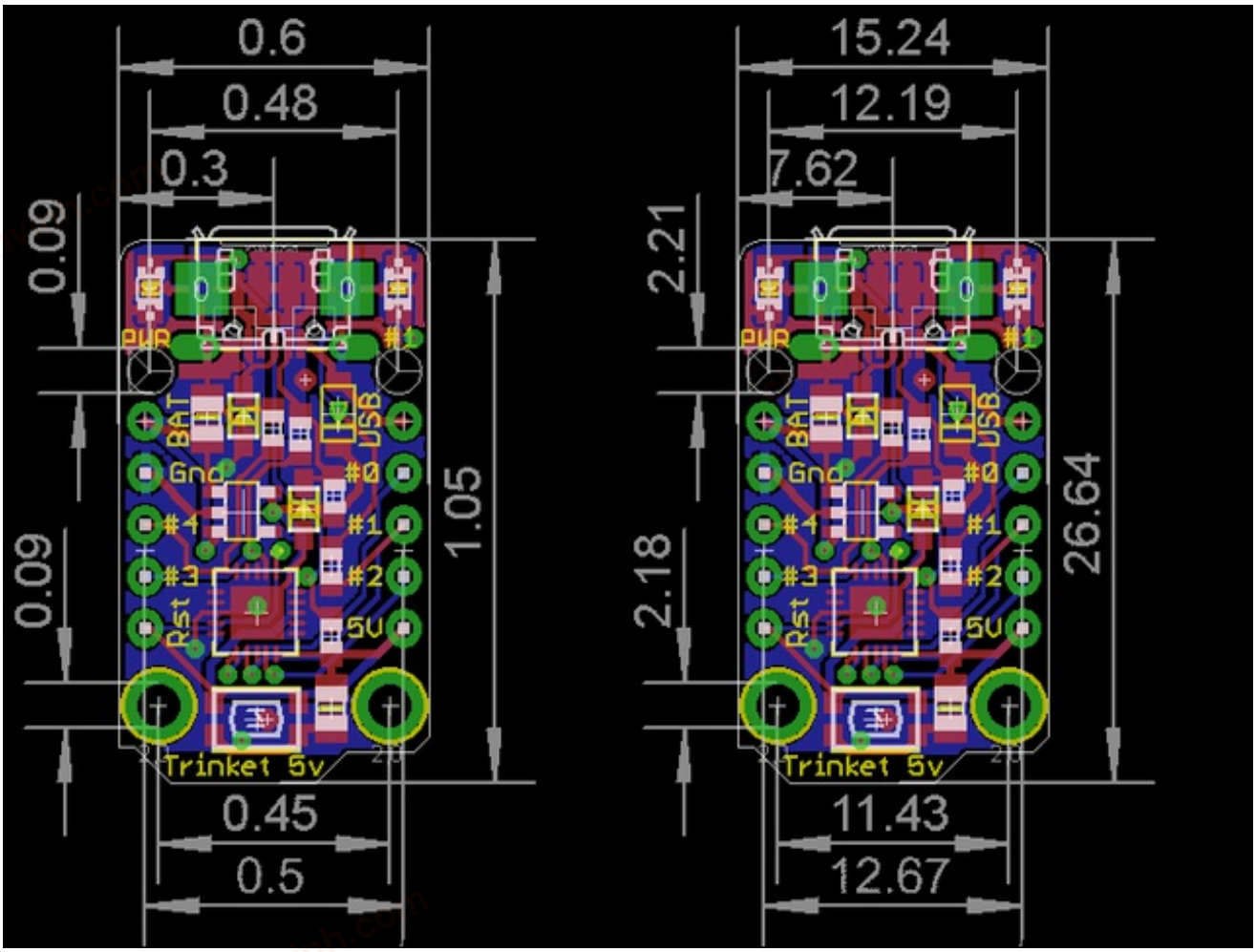


Newer Trinket (Micro USB) dimensions

These are the same for the 3V and 5V versions

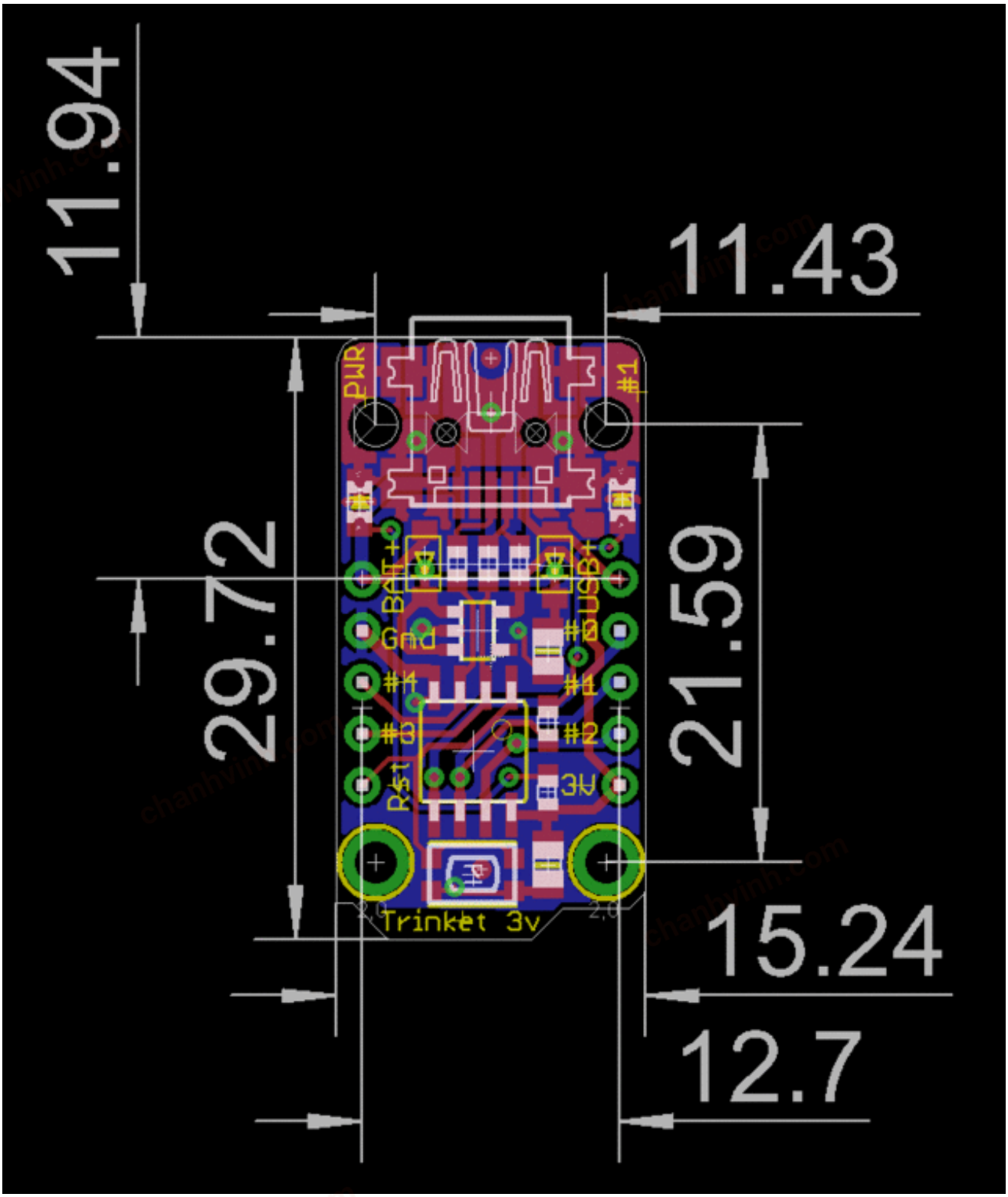
The following shows measurements in mm, both version of the Trinket have the exact same dimensions for hole placement & outline

For more dimensions or if something is missing you need, check the Eagle CAD files above which will give you the location of every component and hole!



Older Trinket (Mini USB) dimensions

These are the same for the 3V and 5V versions


11.94
11.43
29.72
21.59
15.24
12.7
PWR
#1
Bat+
USB+
Gnd
#0
#4
#1
#3
#2
Rst
3V
Trinket 3v

FAQ

How come I can't find the Trinket Serial (COM) Port? Why is no Serial port found when the Trinket is plugged in?

Trinket (and Pro Trinket) **do not** have a USB-serial converter chip on board, this is how we can make them so small! Instead of a serial console, the Trinket is programmed *directly* over USB. No COM/Serial port is used at all!

I'd like to use Trinket with Linux....
We don't guarantee Linux support since Linux varies from distro to distro,but here's a very nice tutorial about Trinket with Ubuntu 14) (http://adafru.it/dML)
I can't seem to upload to my Trinket when it's plugged into a USB 3.0 port (newer Macbooks have USB 3 ports)
Trinket's bootloader is finicky about USB 3 ports, and might not work on them. Try connecting to any USB 2 ports you have or go through a USB 1 or USB 2 hub (nearly all hubs are v2 or v1 instead of v3)
My Trinket 16MHz can't control servos, NeoPixels, but the code is uploading OK?

Sounds like the Trinket thinks its running at 8MHz but the Arduino software thinks it's running at 16MHz, this causs timing-specific stuff like Servos and NeoPixels to not work. Don't forget you **must turn on 16MHz speedup** if you are uploading to "Trinket 16MHz"

Do so by adding

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

At the very top of your sketch, and then

```
if (F_CPU == 16000000) clock_prescale_set(clock_div_1);
```

first thing in **setup()**

See more details here (http://adafru.it/jAd)

Hmm I'm still having problems with Avrdude - and I definitely did the required updates

One fix that works for some people is to edit**avrdude.conf** and set the

```
chip_erase_delay = 900000;
```

under the **ATtiny85** heading to

```
chip_erase_delay = 400000;
```

That is, a shorter delay.

Can Trinket driver Neopixels (strips, squares, etc)? How many?
Yes! Trinket was designed to drive short segments of NeoPixels. There is enough RAM on the attiny85 to drive 100 pixels, but depending on program RAM usage you may have to scale back to 60 or 40.

You can use EITHER the 3V or 5V Trinket, at EITHER 8 or 16MHz!

To use with neopixels:

1. Connect the + power line of the strip to VBUS (5V from USB), to VBAT if you are powering the Trinket with 4-7VDC, or to a separate 4-7VDC power source such as a 3 or 4 pack of AA batteries.
2. Connect the - common ground to the battery pack (if being used) and also to the Trinket GND pin
3. Connect the data in line to Trinket #1 - this will let you also see when data is being sent because the #1 red LED will flicker. You can use other pins but start with #1 since its easiest to debug and use
4. Install the NeoPixel library as detailed in our Uber Guide (http://adafru.it/cEz), and change the **PIN** to **1** (its 6 by default)
5. Upload and enjoy!

Can the Trinket drive your Adafruit I2C LED Backpacks for 7-segment/matrix displays?
Short answer: yes! Check out http://learn.adafruit.com/tap-tempo-trinket (http://adafru.it/cEA) for a tutorial on driving the 7-segment displays. Long answer: we think there's not enough space for all of the fonts for the 8x8 so you might be able to drive the 8x8 matrix in 'raw' mode (see the HT16K33 example sketch in the LEDBackpack Library) but unfortunately not with built-in font support.

That tutorial also shows how to use the TinyM I2C driver, which works great on the ATtiny85, and adapt other existing libraries for the Trinket
Can Trinket drive a Servo?
Yup! In fact you can use 3 servos as long as they are powered by a good 5V supply, check out this guide for more details (http://adafru.it/cFC)
Why does Windows sound the Connect/Disconnect chimes every ten seconds?
The Trinket only appears to be a USBtinyISP device when the bootloader is running. By design, the bootloader only runs for 10 seconds and then jumps back to the main user sketch. this causes the 'disconnect' sound.

On a new trinket, the main sketch will automatically jump back to the bootloader, which will

then cause the 'connect' sound. This cycle will repeat until a user sketch is loaded.

This situation can also happen if you load a sketch with a bug in it that causes a CPU reset.
Can I power the directly on the 5.0V or 3.3V (e.g. treating those as a power input)?

Technically yes - you can provide regulated 5V or 3.3V directly to the lower right hand power pin. However, the regulator dropout is quite low, its safer and better to power the VBAT pin since that way you wont be 'fighting' the USB port if you plug in USB