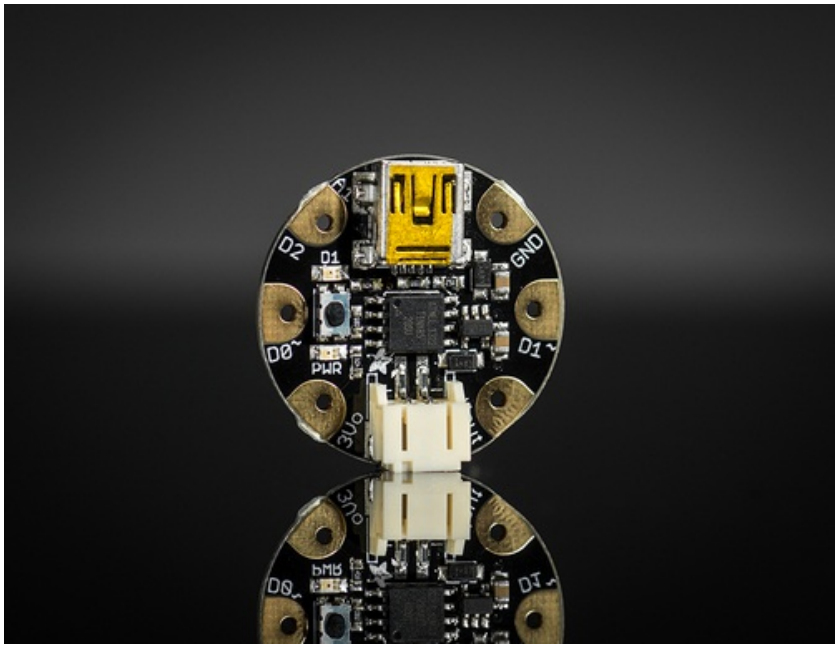
adafruit learning system

Introducing Gemma

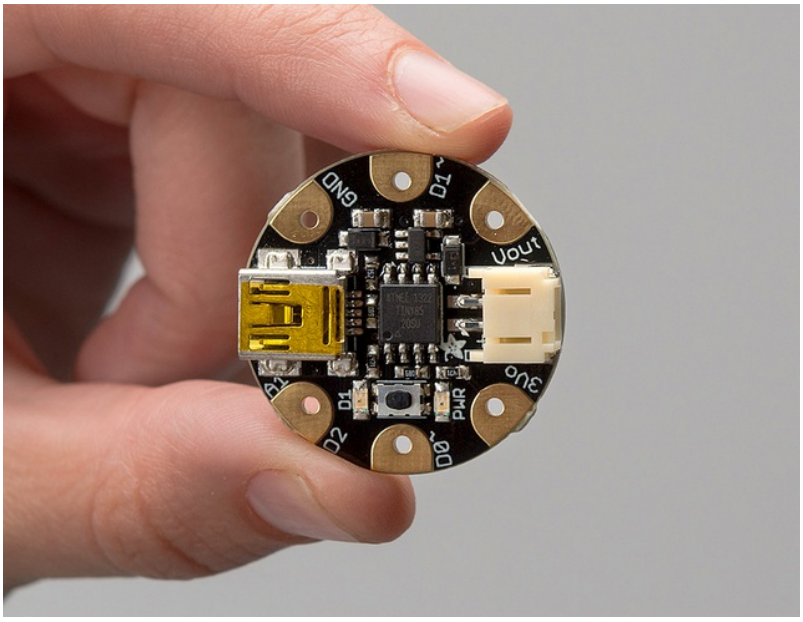
Created by Ladyada



Last updated on 2013-12-08 05:00:41 PM EST

Guide Contents

Introduction



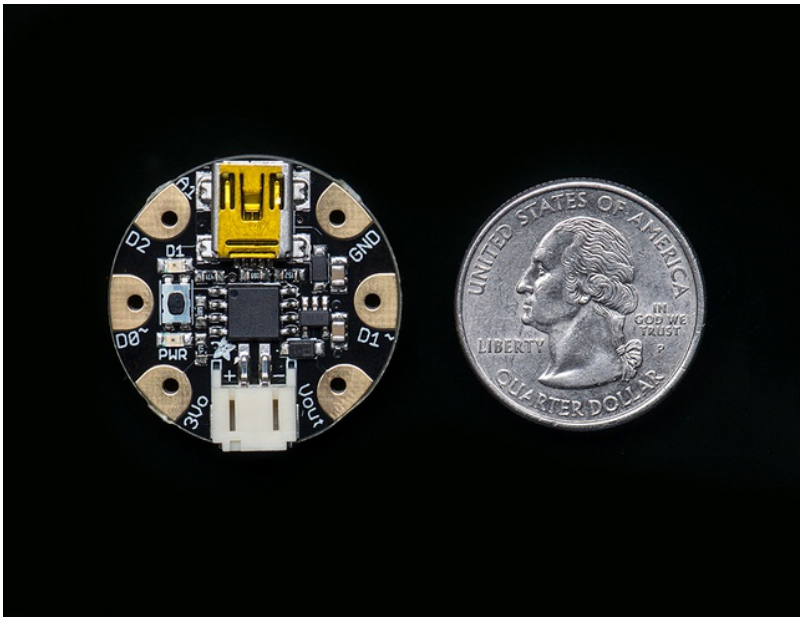
Love Flora but want a bite-sized version? Look no further, Gemma is a tiny wearable platform board with a lot of might in a 1" diameter package. Powered by a Attiny85 and programmable with an Arduino IDE over USB, you'll be able to realize any wearable project!

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 Flora and you aren't willing to take apart the project you worked so hard to design. It's our lowest-cost sewable controller!

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 (it uses 2 of the 5 I/O pins, leaving you with 3). In fact we even made some simple modifications to the Arduino IDE so that it works like a mini-Flora. Perfect for small & simple projects the Gemma will be your go-to wearable electronics platform.

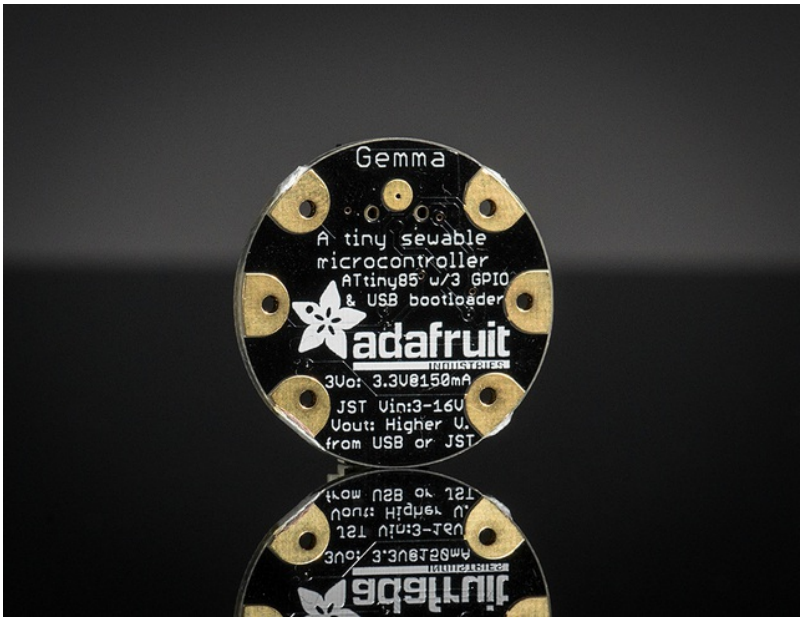
Even though you can program Gemma using the Arduino IDE, it's not a fully 100% Arduino-compatible. There are some things you trade off for such a small and low cost microcontroller!

- Gemma 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 Gemma's bootloader. Simply use a USB v2 port or a USB hub in between
- Gemma is not supported on Linux operating system at this time - try Mac OS or Windows!

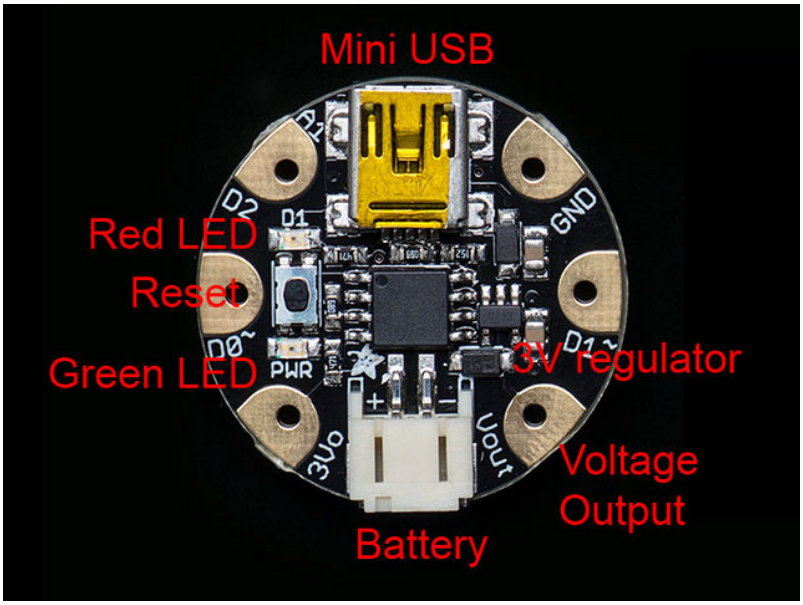


Here are some useful specifications!

- Super small, only 1.1" / 28mm diameter and 0.28" / 7mm thick.
- Easy-to-sew or solder pads for embedding in your wearable project
- Low cost enough, you can use one for every weekend project
- ATtiny85 on-board, 8K of flash, 512 byte of SRAM, 512 bytes of EEPROM
- Internal oscillator runs at 8MHz
- Ultra low power, draws only 9 mA while running
- USB bootloader with a nice LED indicator looks just like a USBtinyISP so you can program it with the Arduino IDE (with a few simple config modifications)
- Mini-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
- ~5.25K bytes available for use (2.75K taken for the bootloader)
- 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.
- 3 GPIO - The 3 independent IO pins have 1 analog input and 2 PWM output as well.
- Hardware I2C capability for breakout & sensor interfacing.


Gemma
A tiny sewable
microcontroller
ATtiny85 6/3 GPIO
& USB bootloader
adafruit
INDUSTRIES
3Vo: 3.3V@150mA
JST Vin:3-16V
Vout: Higher V.
from USB or JST

Guided Tour

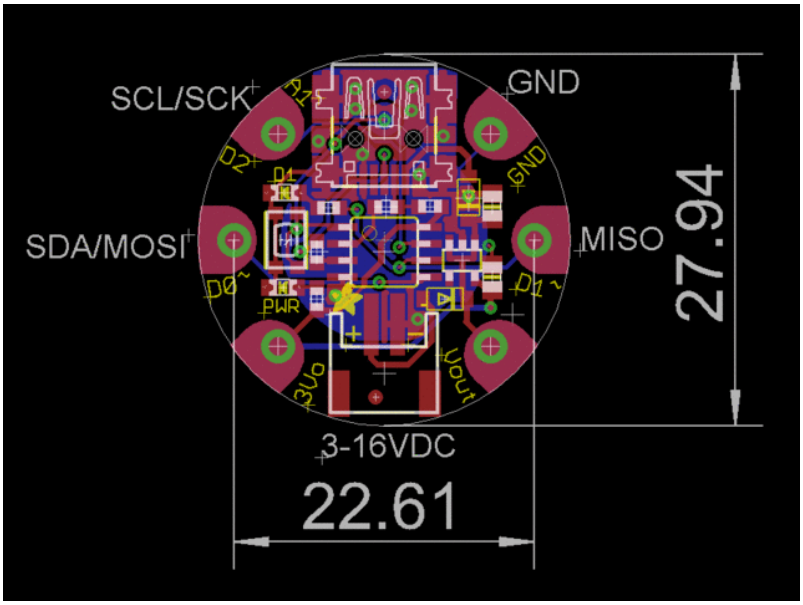


Let me take you on a tour of your Gemma! Each Gemma is assembled here at Adafruit and comes chock-full of good design to make it a joy to use.

- **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.
- **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 Gemma is in bootloader mode, and its also handy for when you want an indicator LED.
- **JST Battery Input** - take your Gemma 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. Works great with a Lithium Polymer battery or our 3xAAA battery packs with a JST connector on the end
- **Voltage Output** - This pin will give you either the battery power or USB power, whichever has a higher voltage. Its great when you want to power something like NeoPixels, that might use more than the 150mA available from the onboard regulator
- **3V Regulator Out** - The on-board voltage regulator can supply up to 150mA at a steady 3.3V from up to 16VDC
- **Sewing friendly pads** - You can easily sew to these pads, and they're gold plated so they wont corrode (oxidize). You can also use alligator clips or solder directly to them.

- **GPIO!** - 3 GPIO pins, at 3V logic, check the next section for a detailed pinout guide
- **Reset Button** - an onboard reset button will launch the bootloader when pressed and the Gemma is plugged into a computer. If it is not connected to a computer, it's smart enough to go straight to the program.

Pinouts



This diagram shows the physical size of the Gemma (diameter in mm) and the distance from pad to pad. The pads are at exact 90 & 45 degree angles except for 0 and 180 which is where the USB/JST connectors go

JST Battery Input

There is no battery INPUT pin on the Gemma. You can connect a battery via the JST jack. We have found that Lipoly batteries (http://adafru.it/cFB), coin-cells (http://adafru.it/783), and AAA's (http://adafru.it/727) work great. You can also make your own battery input pack using a plain JST cable (http://adafru.it/261). And use a JST extension cable if necessary (http://adafru.it/1131).

You can plug anything from around 4VDC up to 16VDC, but we suggest 4-6V since higher voltages just get wasted as heat. This input is polarity protected. If the green PWR LED lights up, you're good to go. There is no off switch on the Gemma, so unplug or switch off the battery pack when done.

Power Pads

Half of the pads on the Gemma are related to power in and out: **3Vo** , **Vout** and **GND**

- **Vout** - This is a voltage **OUTPUT** pin, it will be connected to *either* the USB power or the battery input, whichever has the higher voltage. This output does not connect to the regulator so you can draw as much current as your USB port / Battery can provide (in general, thats about 500mA)

- **3Vo** - This is the **3.3V OUTPUT** pad from the voltage regulator. It can provide up to 150mA at a steady 3.3V. Good for sensors or small LEDs or other 3V devices.
- **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

Input/Output Pads

Next we will cover the 3 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 Gemma, the GPIO are 3.3V output level, and should not be used with 5V inputs. In general, most 5V devices are OK with 3.3V output though.

The 3 GPIO 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

- **Pad #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.
- **Pad #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).
- **Pad #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.

Secret Reset Pad

On the off chance you want to reprogram your Gemma with an AVR burner, the bottom of the board has a large pad that is connected to the Reset pin. We use it for testing and you will likely never need it but it is there if you do.

About the Bootloader

About the bootloader

One of the challenges with the Gemma 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-Gemma products or projects. Purchase a USB VID for yourself at http://www.usb.org/developers/vendor/

Gemma USB Drivers for Windows

The cool thing about the bootloader on the Gemma is it just looks like a classic USBtinyISP AVR programmer. This makes it easy to use with AVRdude or Arduino IDE with only minor configuration changes. Before you start, you may need to install the USBtinyISP USB drivers

Drivers are only required for Windows, if you are using a Mac or Linux, drivers are not required!

For details on installing the drivers for Windows XP, 7, 8 etc... please read this page! (http://adafru.it/cDY)
Don't forget for Windows 8 you will have to turn off driver signing temporarily in order to allow the USBtiny/Trinket driver to be installed

If you're good at installing drivers, you can just click here to download the ZIP (http://adafru.it/cnK)

Don't forget to plug in the Gemma via a known-good USB cable to start the process. You should see the green power LED lit and the red bootloading LED pulse indicating that the Gemma is ready to start programming. If you've programmed the Gemma since getting it, you can always get it back to the bootloader state by pressing the small onboard reset button.

Special Notes on using Gemma with Linux

Gemma 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 Gemma 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. The board shown is a Trinket, which uses the same upload system as Gemma.

Setting up with Arduino IDE

Chances are, you picked up a Gemma because it is programmable with the Arduino IDE. Note that the Gemma is not a full Flora or Arduino-compatible, it uses a different (smaller) chip than the Flora, 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 Gemma 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!

The Fast Way

If you don't want to modify an existing Arduino IDE install, you can simply download our ready-to-go and tested Flora/Gemma/Trinket-ified v1.05 right here:

Mac Arduino IDE v1.05 w/Trinket, Gemma, Flora

http://adafru.it/cTN

Windows Arduino IDE v1.05 w/Trinket, Gemma, Flora

http://adafru.it/cUa

If you're using Mac OS Mavericks you will need to update the setting to permit running Arduino IDE

1. Go to your Security and Privacy Settings
2. Click the Lock Icon and Login
3. Change "Allow Apps Downloaded From": to "Mac App Store and identified developers"

The Slow Way

Step 0. Install Arduino IDE

Gemma support is not native to the Arduino IDE but, luckily, adding it takes only a few minutes and you only have to do it once! This tutorial will base the IDE off of v1.0.5 which is current at time of writing. You can try later versions but v1.0.5 is at least guaranteed to work

Adding Gemma support does not affect any other boards that are affected so you can continue to use the IDE with any Arduino board currently supported.

Step 1. Add ATtiny85 Support

Download the following file by pressing the button.

Download Trinkethardwaresupport.zip

http://adafru.it/cEh

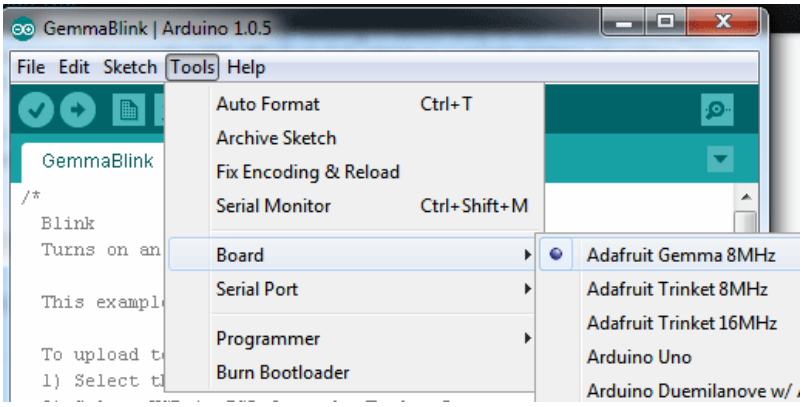
Unzip it and move the **hardware** folder from the zip file and place it into to your Arduino *sketchbook* folder. Your sketchbook folder is the folder where the Arduino IDE stores your sketches. This folder is automatically created by the IDE when you install it. If this is your first time using the Arduino IDE, it will be empty!

On Linux machines, the folder is named "Sketchbook" and it is typically located in /home/[username]

On Windows and Macintosh machines, the default name of the folder is "Arduino" and is located in your Documents folder.

> This is a common source of confusion on Windows and Mac machines, your sketchbook folder is not named "sketchbook" it is named "Arduino"!

Now you can start (or restart the the IDE) and check the **Tools->Board** menu, you should see the three new entries for Trinket and Gemma:



OK you are half done! Next is updating the avrdude configuration file.

Step 2. Updating avrdude.conf

The second step is to update the AVR chip program upload helper to be a little more patient with the ATtiny85 bootloader we have on the Gemma. We will update the description of the chip's erase cycle to be longer, to avoid timeouts and errors.

Windows and Linux users can download the new **avrdude.conf** by clicking this button:

avrdude.conf for Windows & Linux

http://adafru.it/cE3

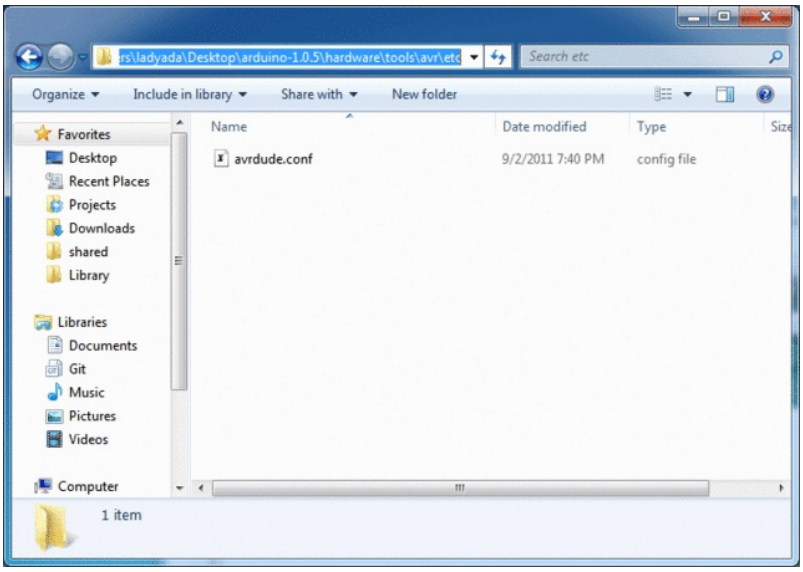
Mac users should download this version:

avrdude.conf for Mac

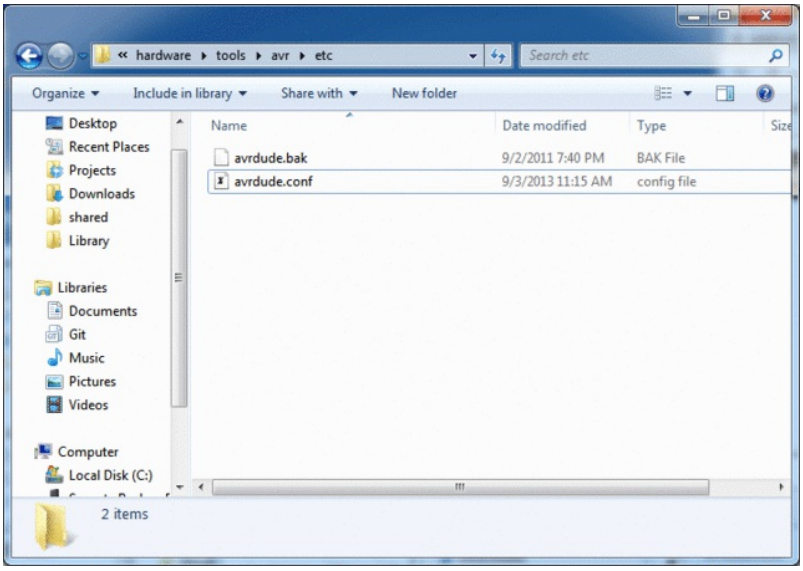
http://adafru.it/cEy

Now we will hunt for the original **avrdude.conf** file. If you are using a Mac, right-click on the Arduino application icon and select “Show Package Contents.” On Windows you will have to find the installation directory, which may be a folder on the Desktop or possibly in **C:\Program Files** if you used the installer. Likewise in Linux it is where-ever you uncompressed the folder.

Now find the **hardware/tools/avr/etc** folder (on Mac it's Contents/Resources/Java/hardware/tools/avr/etc), and inside you should see the **avrdude.conf** file. You can also use your operating system's **find** tool to locate it.



Rename the old **avrdude.conf** to **avrdude.bak** and copy over the *new* **avrdude.conf** to the same folder

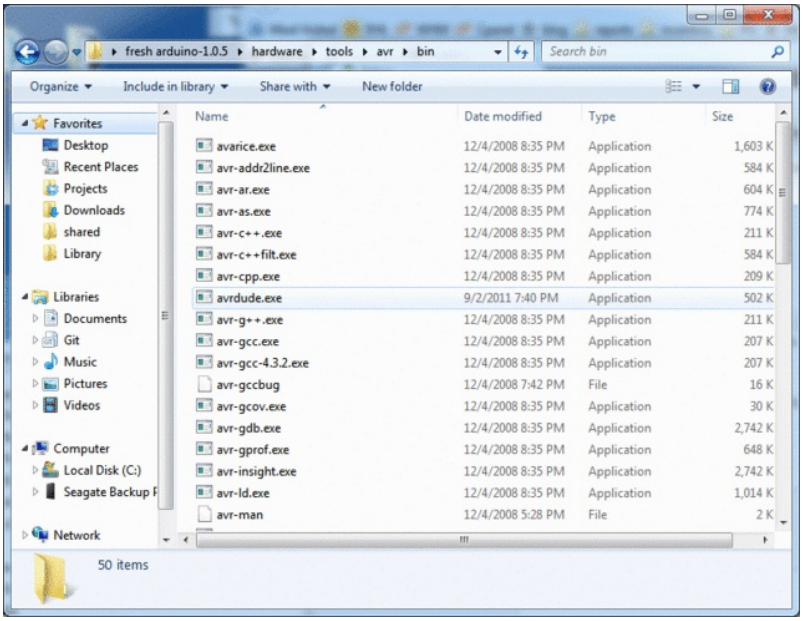


Step 3. Update 'ld' linker

There's a bug in the 'linker' used by Arduino on Mac & Windows, where you can't make sketches that are larger than 4K on the Attiny85. Since its really likely you'll make sketches this large, we suggest replacing it. It's a lot like replacing the **avrdude.conf**

On Windows: explore the Arduino folder and get to the **hardware\tools\avr\bin** subfolder, you'll see a lot of files starting with **avr-xxx**.

On Mac: Explore the App and find **Adafruit Arduino.app/Contents/Resources/Java/hardware/tools/avr/avr/bin**



Download the **ld** for your operating system and replace the one you have now (thanks to TCWorld for this fix! (http://adafru.it/cGP))

Download the new Id for Windows

http://adafru.it/cGQ

Download the new Id for Mac

http://adafru.it/cGR

Now restart the Arduino IDE. You are done with setup! Now it's time to say "hello, world" to your new Gemma with the basic Blink sketch.

Blink!

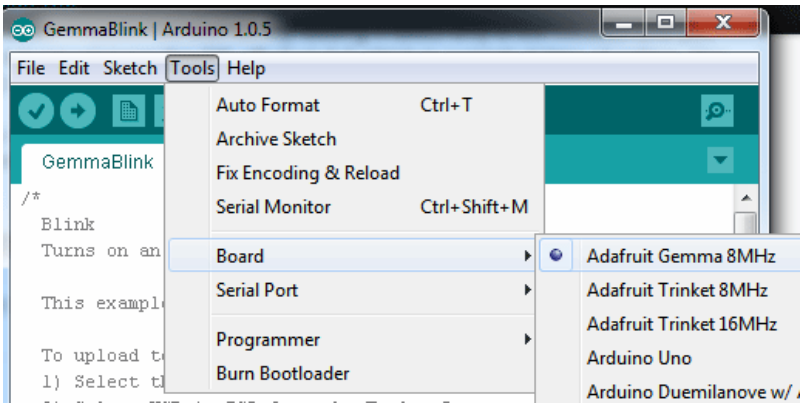
OK let's practice uploading a sketch by uploading the classic Blink sketch. This program with blink the onboard red LED that is connected to pin #1. Create a new sketch and copy&paste the following into it, you can then save it as **gemmablink** or something similar, so you have it handy

> If you are using Linux you will have to be "root" running the Arduino program to have access to the USB port

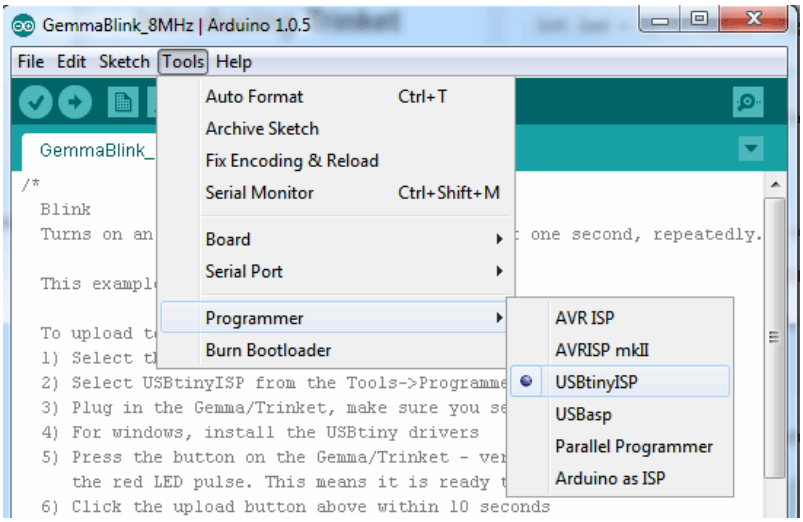
```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  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 **Gemma 8MHz** board from the **Tools->Board** menu

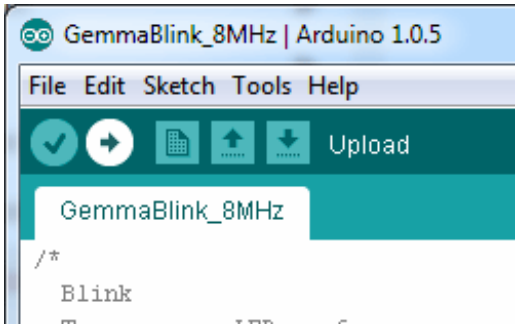


Then, select **USBtinyISP** from the **Tools->Programmer** sub-menu

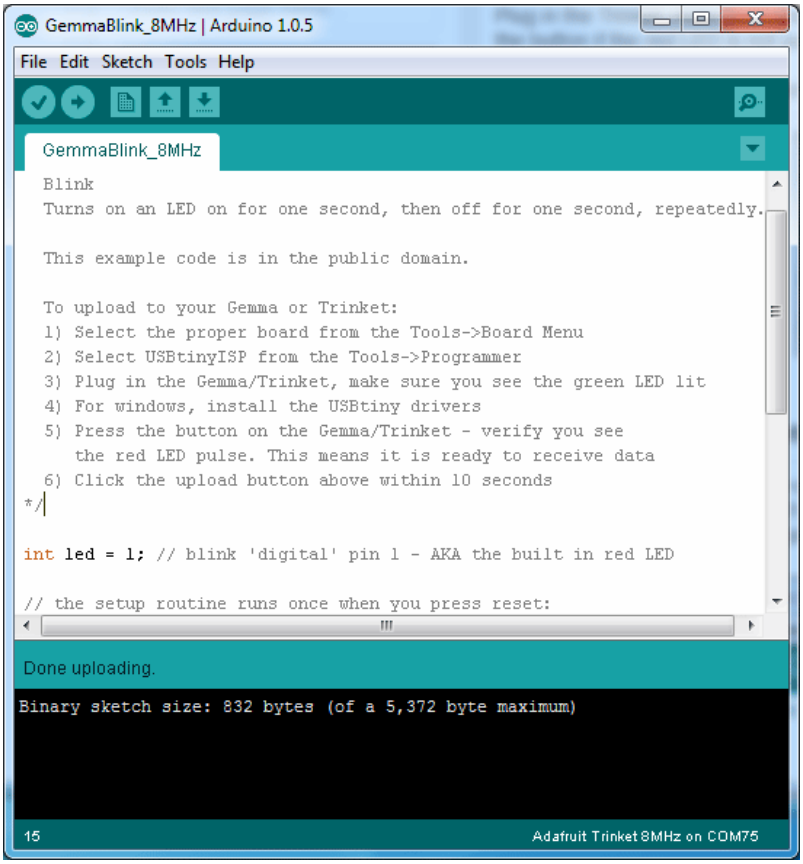


Plug in the Gemma, make sure you see the green LED lit (power good) and the red LED pulsing. Press the button if the red LED is not pulsing, to get into bootloader mode.

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



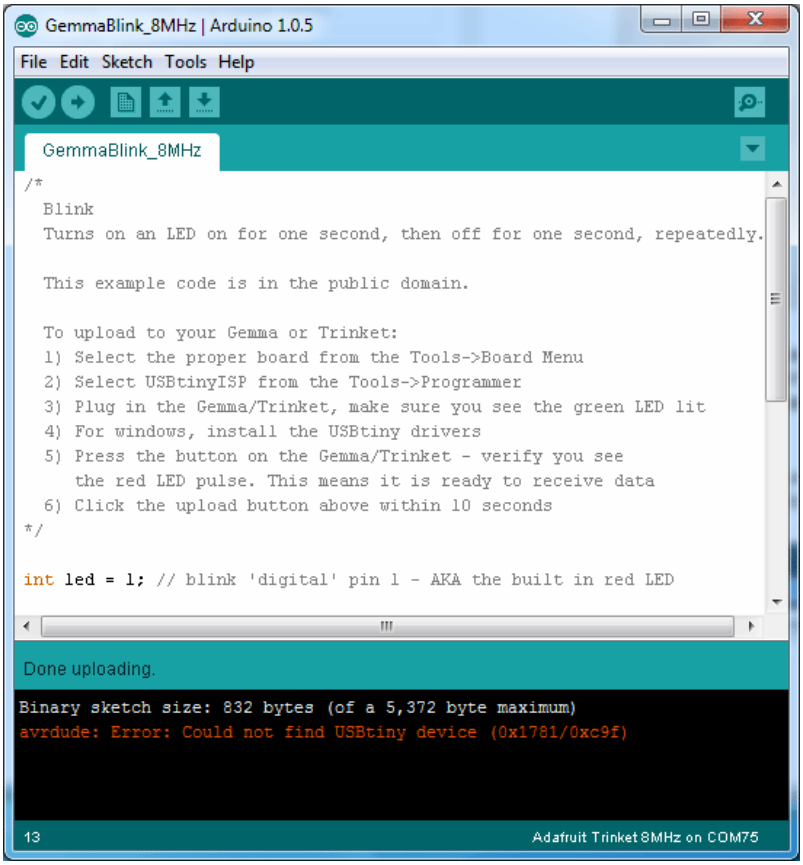
If everything goes smoothly you should see the following (no red error messages) and of course, the red LED on the Gemma will blink on/off once a second



Something Went Wrong!

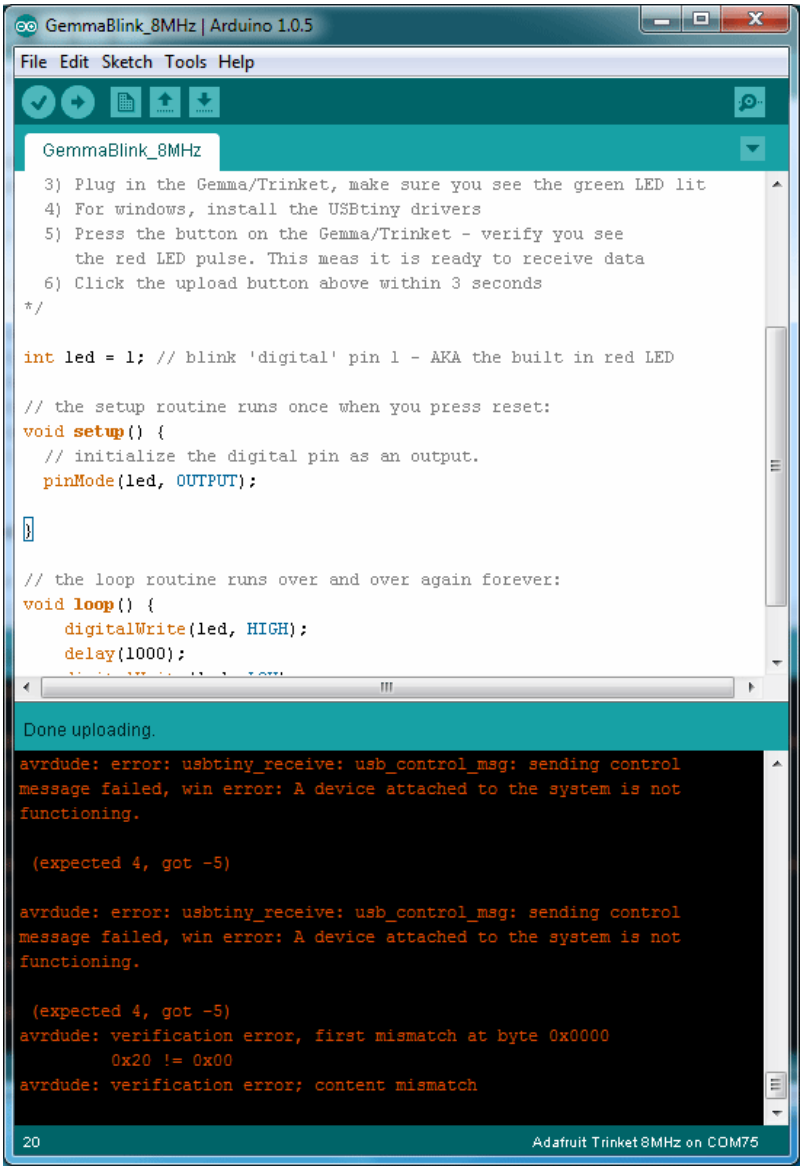
If you get the error message avrdude: Error: Could not find USBtiny device (0x1781/0xc9f)

That means the bootloader wasn't active. Make sure to press the button on the Gemma 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

Check that you updated the avrdude.conf file above - if you don't update the description of the Attiny85 in the configure file by replacing it, the IDE wont know to be patient with the Gemma bootloader and will have many upload errors


GemmaBlink_8MHz | Arduino 1.0.5
File Edit Sketch Tools Help
GemmaBlink_8MHz
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 meas it is ready to receive data
6) Click the upload button above within 3 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);
Done uploading.
avrdude: error: usbtiny_receive: usb_control_msg: sending control
message failed, win error: A device attached to the system is not
functioning.
(expected 4, got -5)
avrdude: error: usbtiny_receive: usb_control_msg: sending control
message failed, win error: A device attached to the system is not
functioning.
(expected 4, got -5)
avrdude: verification error, first mismatch at byte 0x0000
0x20 != 0x00
avrdude: verification error; content mismatch
20
Adafruit Trinket 8MHz on COM75

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 #2 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 **A1**)

For example, to read an analog voltage on pin #2, you would call **analogRead(A1)**

analogWrite()

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

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);
  }
}
```

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!

Downloads

Datasheets

Datasheet for the onboard regulator used (MIC5225 3.3V)

MIC5225-3.3
http://adafru.it/cE4

Webpage for the ATtiny85, the microcontroller used in the Gemma (http://adafru.it/cE5)

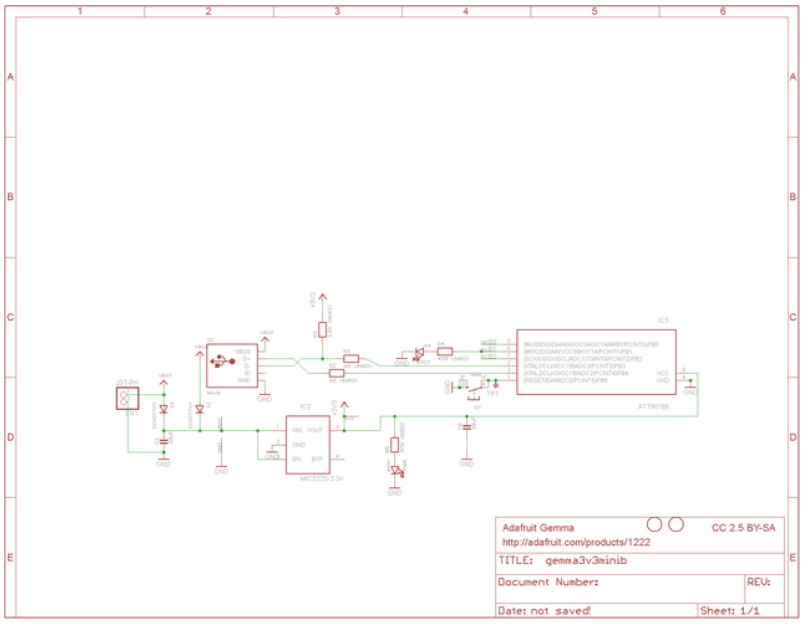
Source code

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!

> 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

Gemma Schematic:



FAQ

When uploading with the Arduino IDE, I get a lot of "(expected 4, got -5)" warnings and then "avrdude: verification error; content mismatch"

Check that you followed the instructions for updating the Arduino IDE, including replacing the old avrdude.conf (http://adafru.it/cEY)- this step is not optional!

Check that you followed the instructions for updating the Arduino IDE, including replacing the old avrdude.conf (http://adafru.it/cEY)- this step is <u>not optional!</u>

Can Gemma drive Neopixels (strips, squares, etc)? How many?

Yes! Gemma 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.

To use with neopixels:

1. Connect the + power line of the strip to **Vout** on the Gemma, 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 Gemma **GND** pin
3. Connect the data in line to Gemma #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!

Yes! Gemma 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.

To use with neopixels:
 Connect the + power line of the strip to Vout on the Gemma, or to a separate 4-7VDC power source such as a 3 or 4 pack of AA batteries. Connect the - common ground to the battery pack (if being used) and also to the Gemma GND pin Connect the data in line to Gemma #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 Install the NeoPixel library as detailed in our Uber Guide (http://adafru.it/cEz), and change the PIN to 1 (its 6 by default) Upload and enjoy!

Can Gemma use the Flora Lux/Compass/Accelerometer/Color/GPS Sensors?

Maybe! We think we can get Gemma working with some of the basic sensors, but at this moment we don't have tutorials or examples, Flora code will not compile directly for the

Gemma since the processors are different. Right now we think Gemma is best suited for basic buttons/LEDs/Neopixels type stuff

Maybe! We think we can get Gemma working with some of the basic sensors, but at this moment we don't have tutorials or examples, Flora code will not compile directly for the Gemma since the processors are different. Right now we think Gemma is best suited for basic buttons/LEDs/Neopixels type stuff

Can Gemma 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 Gemma/Trinket

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 Gemma/Trinket

Can Gemma 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)

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)

Gemma runs at 8MHz, but I really need it to run at 16 MHz, is this possible?

It is possible to run the Gemma at 16MHz, but the processor is not specified for 16MHz at 3.3V logic so it is considered overclocking!
However, the AVR series is pretty forgiving for overclocking, so *you may be able to run the 3V Gemma 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 if you can get away with it!

To run at 16Mhz, use the Trinket 16Mhz board definition and modify your sketch as described here. (http://adafru.it/cFD)

It is possible to run the Gemma at 16MHz, but the processor is not specified for 16MHz at 3.3V logic so it is considered overclocking!
However, the AVR series is pretty forgiving for overclocking, so you may be able to run the 3V Gemma 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 if you can get away with it!

To run at 16Mhz, use the Trinket 16Mhz board definition and modify your sketch as described here. (http://adafru.it/cFD)
