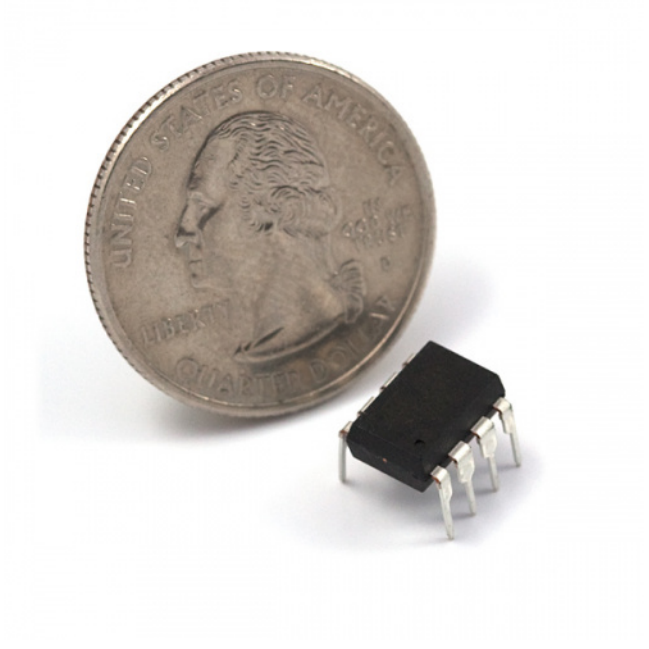
Tiny AVR Programmer Hookup Guide a learn.sparkfun.com tutorial

Available online at: http://sfe.io/t150

Contents

Introduction

Arduino is awesome. The boards are solid, the programming language and IDE are easy, and the community is awesome. But for a lot of electronics projects, an Arduino is overkill. If you're just blinking a few LEDs, and reading a single sensor, you can get the job done **smaller and cheaper** using a simple IC, like the ATtiny85.



AVR 8 Pin 20MHz 8K 4A/D - ATtiny85

COM-09378
$3.50
8
Favorited Favorite 47
Wish List

Our hero! The ATtiny85.

Unfortunately, the ATtiny85 doesn't have a well-known, ubiquitous development platform like Arduino's Uno or Leonardo. And 8kB of program space doesn't leave much room for a bootloader, so an extra programmer is usually required. On top of that, standard Arduino doesn't support the chip. That doesn't mean programming the ATtiny85 in Arduino isn't possible, though! Enter the Tiny AVR Programmer...

Tiny AVR Programmer

PGM-11801
$17.50
101
Favorited Favorite 74
Wish List

The Tiny AVR Programmer is a general AVR programmer, but it's specifically designed to allow **quick-and-easy programming** of ATtiny85's (as well as 45's) compared to the pocket AVR programmer. It has an on-board socket, where the little 8-pin IC can be plugged in and directly programmed. No messy wires or soldering required! Once you've programmed the ATtiny85, just remove it from the Programmer, and stick it into a breadboard or prototyping board.

The Tiny AVR Programmer can also be used as a general purpose AVR programmer. It can directly program almost all AVR's (including the ATmega328 and ATmega32U4) whether they're on Arduino boards or in a breadboard.

Covered In This Tutorial

In this hookup guide, we'll show how you can program ATtiny85's using the Tiny AVR Programmer and **Arduino**. We'll cover everything from driver installation to Arduino programming tips.

Required Materials

In addition to the Tiny AVR Programmer, you'll also need the following items to follow along with this tutorial:

- ATtiny85 -- To be programmed by the programmer. *Remember, you can also use this to flash other AVR chips like the ATtiny84!*
- A computer or laptop with:
  - A free **USB Port**. A USB hub should work too.
  - **Arduino IDE** installed.
- *Optional*:
  - USB Extension Cable -- If your USB port is out of reach, this may help make the Programmer easier to reach.
  - IC Test Clip - SOIC 8-Pin -- If you are using a surface mount ATtiny, this handy dandy little clip makes it easy to program the microcontroller!
  - Jumper Wires Premium M/F -- Useful if you are connecting the Tiny AVR Programmer to another AVR microcontroller that is not the ATtiny85 or the IC Test Clip.

Suggested Reading

- Installing Arduino -- You'll need Arduino installed for the Programming in Arduino section of this tutorial. There is an ATtiny85 addon for Arduino, which enables you to program the tiny AVRs in the familiar Arduino interface.
- Integrated Circuits -- This tutorial goes over the basic concepts of integrated circuits. The little black chips that the Tiny AVR Programmer is designed to program.
- Polarity -- Specifically the Integrated Circuits section. You should know all about IC notches and dots.

Installing Arduino IDE

A step-by-step guide to installing and testing the Arduino software on Windows, Mac, and Linux.
Favorited Favorite 16

Polarity

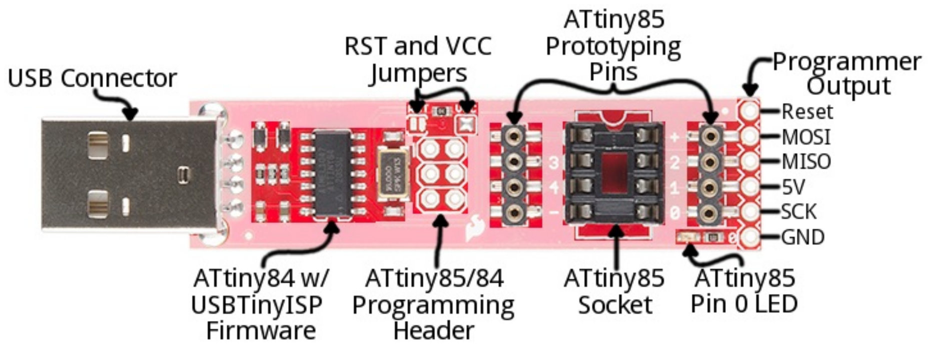
An introduction to polarity in electronic components. Discover what polarity is, which parts have it, and how to identify it.
Favorited Favorite 49

Integrated Circuits

An introduction to integrated circuits (ICs). Electronics' ubiquitous black chips. Includes a focus on the variety of IC packages.
Favorited Favorite 44

Board Overview

The image below provides a quick overview of the components on the Tiny AVR Programmer:

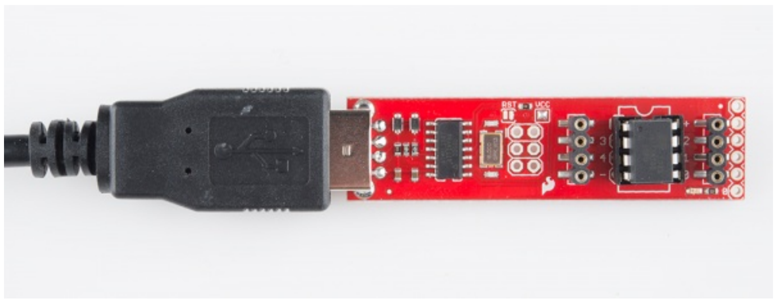


The "brain" of the Tiny AVR Programmer is an **ATtiny84** (not to be confused with the 85), -- the 16-pin surface-mount chip -- which comes preprogrammed with some firmware that makes it look like an AVR programmer. Unless you're writing custom AVR ISP firmware, you shouldn't ever have to mess with this chip. It's a black box. Program data comes into it from your computer, over USB, and it spits out the proper sequence of bytes to load that program into your ATtiny85.

In this tutorial, we'll mostly concern ourselves with the components on the right half of the board. The ATtiny85 programming socket, pin 0 LED, and prototyping pins.

ATtiny85 Socket and Prototyping Pins

The socket and the pins broken out to the sides are what make the Tiny AVR Programmer unique. The 8-pin socket fits both the ATtiny85 and the ATtiny45 DIP packages. Just plug your IC-to-program into this socket, and a-programming you will go!

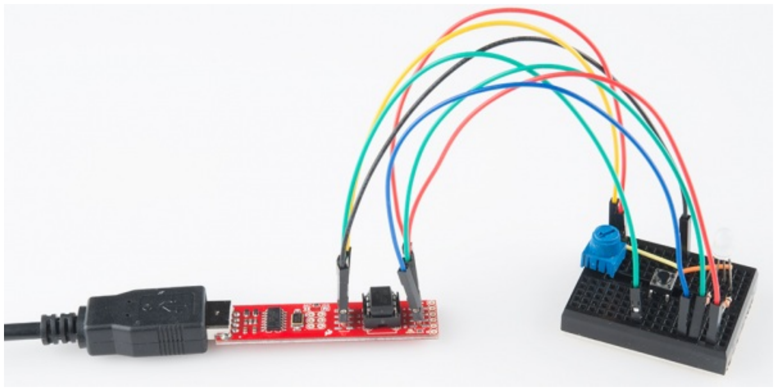


A Tiny AVR Programmer with an ATtiny85 inserted.

When plugging your ATtiny into the socket, take note of the **notch** on both the socket and the white silkscreen on the PCB. This should match the **polarity** of the ATtiny85. Usually the ATtiny85 has a **dot** next to pin 1 of the IC, this should be placed up **towards the notch**.

The +, -, and numerical **labels** on the side of the socket reference the pin numbers and voltage supply inputs of the ATtiny85. These pin numbers can be called in the Arduino IDE as we'll show later in this tutorial.

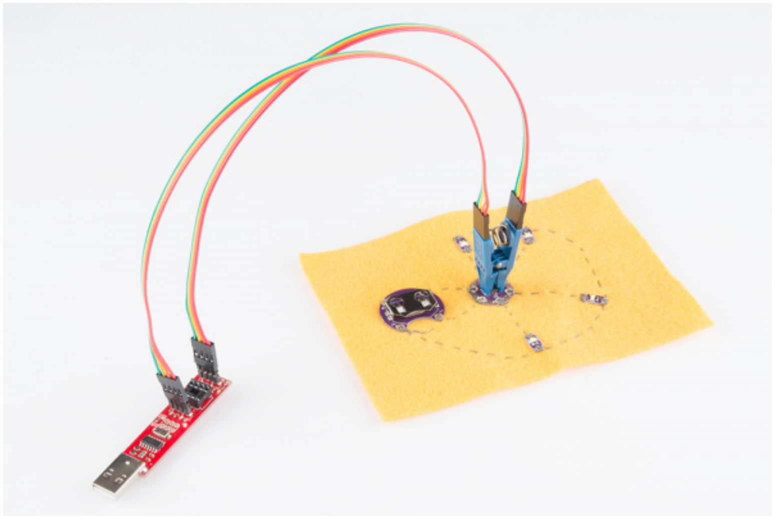
The 4-pin headers on either side of the socket help for prototyping the ATtiny85 out to external circuitry. You can easily plug male jumper wires into these pins, which can be routed to breadboards or other prototyping circuits.



An ATtiny85 being prototyped out to a potentiometer (analog input), button (digital input), and RGB LED (analog/digital output).

Note: The ATtiny85's I^{2}C and SPI functionality cannot be used in this method because the pins are still tied to the SPI pins of the ATtiny84 used to program the chip. The IC needs to be removed from the programmer first.

The 4-pin headers can also be used to connect to surface mount ATtiny85's or other AVR microcontrollers that are on breadboards.



Tiny AVR Programmer connected to an surface mount ATtiny85 that was sewn on fabric to reprogram a LilyTiny.

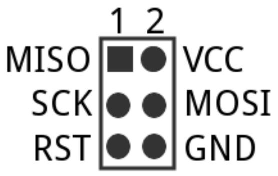
Finally, there's an on-board **amber LED connected to pin 0** of the ATtiny85. This is super-helpful when you're uploading the "Hello, world" blink sketch to an ATtiny85.

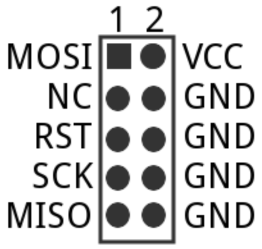
That covers the fundamental stuff on the Tiny AVR Programmer. If you plan on doing more advanced stuff with the board, or just want to know more, feel free to read on. Otherwise, skip ahead to the next page.

Output Programming Pins

The Tiny ISP Programmer is *not* limited to ATtiny85's. It's a full-fledged AVR programmer. This row of six pins can be connected to other AVRs via the standard 2x3- or 2x5-pin ISP headers. You could, for example, connect these pins to your Arduino Uno, Leonardo, etc. to re-flash a bootloader, or upload code using a programmer.

Refer to the pin labels in the image above if you're connecting the Tiny AVR Programmer to another AVR chip. Most AVR development boards break out either a 2x3 or 2x5 programming header, which have the following pin-outs:





Just match up the labels on the Tiny Programmer to the pins on your AVR board/chip, and get ready to program!

The Jumpers

There are two jumpers on the top side of the Tiny AVR Programmer: one is labeled *RST* and the other is *VCC*. Both of these jumpers affect the unpopulated 2x3 ICSP (in-circuit system programmer) header in the middle of the board. Unless you're planning on reprogramming the on-board ATtiny84, these jumpers and pins can generally be ignored.

The ***VCC* jumper** is **normally closed**. It controls the flow of power to the VCC pin on the ICSP header. When closed, power from USB will flow to the ICSP header. When open you'll need to supply power externally to that pin.

The ***RST* jumper** is **normally open**. When closed, this jumper connects the ATtiny84's reset pin to the to the 2x3 programming header. If you ever need to reprogram the ATtiny84 (which, for standard use cases, you shouldn't), you'll have to close this jumper to enable programming it.

Enough talk. Let's start using the programmer. On the next few pages we'll cover driver installation (for Windows users) and show how you can use the Tiny AVR Programmer to program an ATtiny85 in Arduino.

Driver Installation

Before you can start using the Tiny AVR Programmer, you may need to set it up on your computer. If you're using a **Mac or Linux** machine, you **don't need to install drivers**. Just plug the board in, and skip to the Programming in Arduino page.

Tiny AVR Programmer Hookup Guide - Programming in Arduino

If you're using any version of **Windows**, you've got a few steps to follow before you can join your Mac/Linux comrades. There are two sets of instruction for driver installation on this page. The first is the easiest, quickest method, and should work for most everyone. The second installation process is only required if the first one fails -- it takes a more manual approach to the driver installation.

Automatically Install the Drivers with Zadig

To begin, **plug the Tiny AVR Programmer into your computer**. Upon initially connecting the board, Windows will try to automatically install the drivers. Some computers may be lucky, but most will turn up with a message notifying you that the driver install failed.

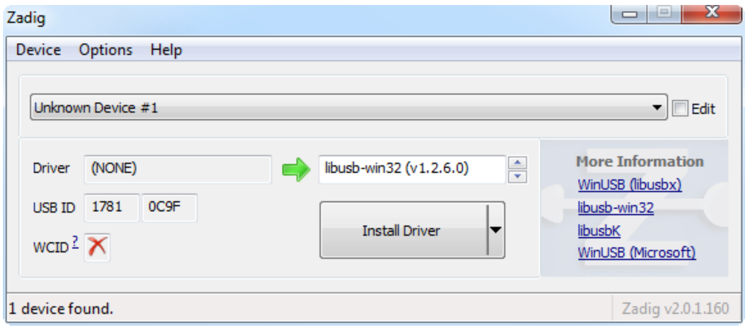
Click the link below to download the Zadig software and drivers:

Download the Zadig USBtiny Drivers (ZIP)

Use your favorite unzipper to extract the ZIP file. Don't forget where you put the extracted folder!

After you've plugged the Tiny AVR Programmer into your computer and your machine has run through the process of checking for and failing to install drivers, proceed to the "**zadig_v2.0.1.160**" folder you just unzipped. Then **Run zadig.exe** software.

Zadig is a wonderful tool that can install the drivers on just about any Windows platform out there. Upon opening the program, you should be greeted with a window like this:



Unknown Device #1

Driver (NONE) → libusb-win32 (v1.2.6.0)

USB ID 1781 0C9F

WCID ?

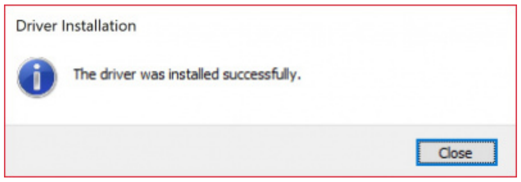
1 device found.

Zadig v2.0.1.160

There are a few options to verify before installing the driver:

- **Select the device** -- The top dropbox controls which device you want to install the driver for. Hopefully you only have one option here, something like "**Unknown Device #1**". If you have more than one option, check your device manager to see if you can make sense of which is which (plugging and unplugging a device usually helps).
- **Select the driver** -- Click the arrows in this box until you happen upon **libusb-win32 (vx.x.x.x)**, that's the driver we want to install.

After verifying those two selections, **click "Install Driver"**. The installation process can take a few minutes, but after you've watched the scroll bar zoom by countless times, you should be greeted with a "**The driver was installed successfully**" message.



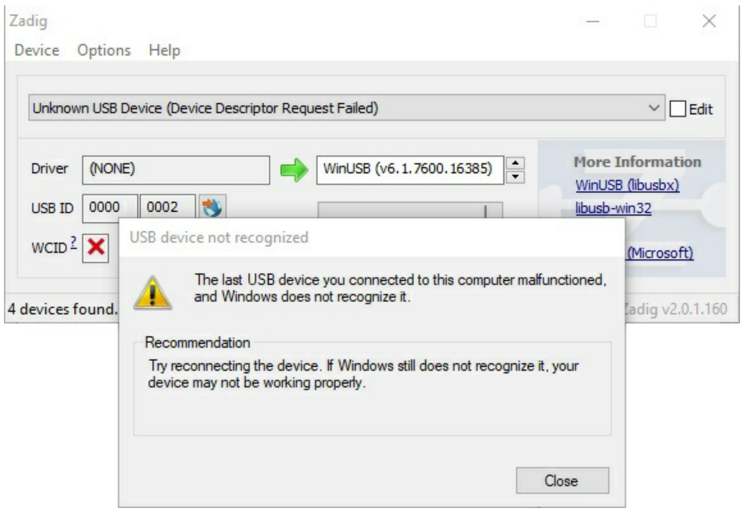
Driver Installation

The driver was installed successfully.

Zadig Driver Installation Issues

After installing the drivers, your computer may respond by indicating that the device was not installed correctly. Here are two methods of troubleshooting driver issues when installing with Zadig.

Troubleshooting Tip: In this case, the *WinUSB* drivers were selected instead of the *libusb-win32* drivers. To remedy the issue, simply go through the guide again to reinstall the correct *libusb-win32* drivers.



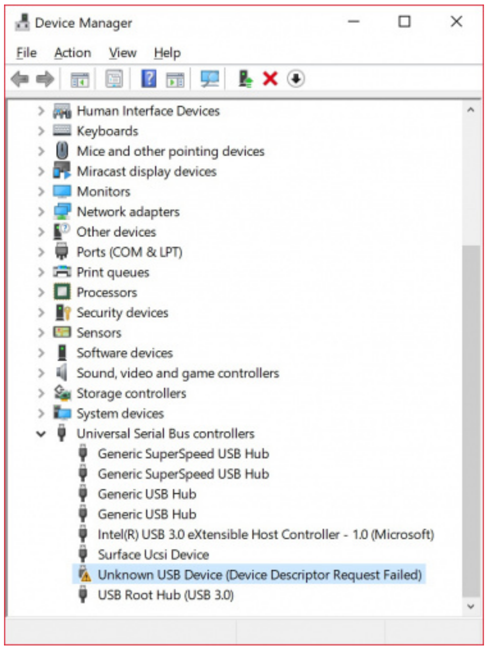
USB device not recognized

The last USB device you connected to this computer malfunctioned, and Windows does not recognize it.

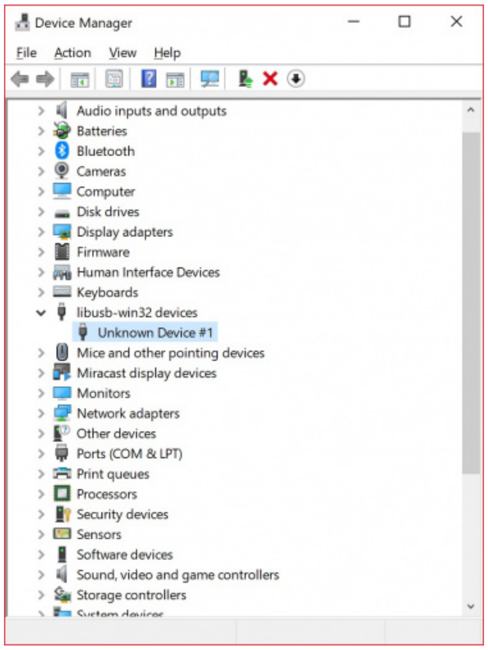
Recommendation

Try reconnecting the device. If Windows still does not recognize it, your device may not be working properly.

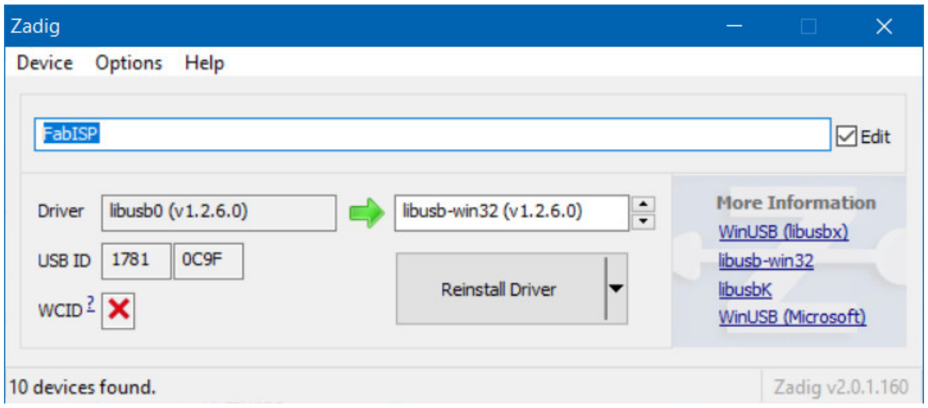
Troubleshooting Tip: In other cases, it may also initialize somewhere in your device manager as an **Unknown USB Device (Device Descriptor Request Failed)** even if you installed the correct drivers:



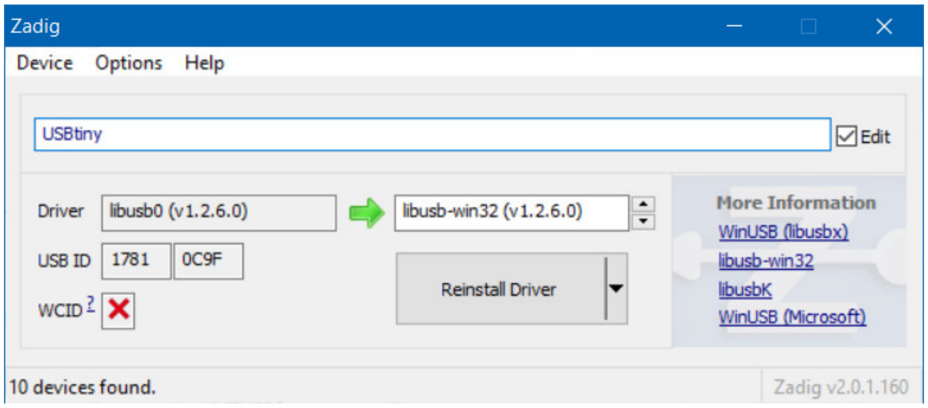
Try unplugging and replugging the Tiny AVR Programmer back into your USB port. Or switch out your USB extension cable for a known good. In some cases, your Tiny AVR Programmer may shows up under the **libusb-win32 devices** as an **Unknown Device #1**. If that is the case you should be good to go!



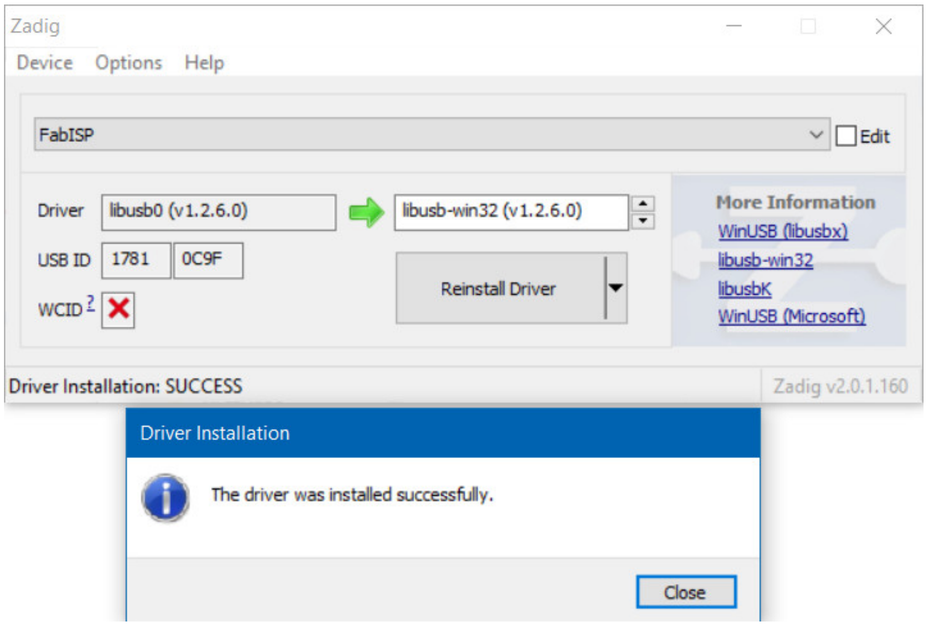
Well done! You've successfully installed the drivers on your computer. However, the driver still shows up as an**Unknown Device #1** (in some cases like the image below, the Tiny AVR Programmer may show up as **libusb-win32 devices** > **FabISP**). But you know what it is! You can use the Zadig software to rename the USB port if you desire. With your programmer connected to your computer and the software open, navigate to the programmer's port. Select the checkbox next to **Edit**.



Type in the name for your port. It can be "**USBtiny**" or in this case. Make sure that the correct driver is selected.



Click **Reinstall Driver**. The driver will reinstall and you should see the same message that indicates that the drivers were successfully installed. You may need to unplug and replug the programmer to your computer to give it a second to refresh again.

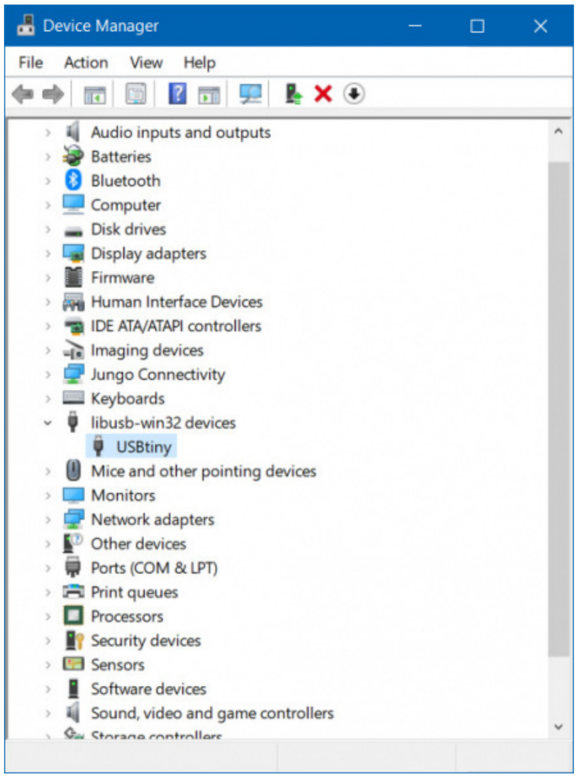


Driver Installation: SUCCESS

Driver Installation

The driver was installed successfully.

Open up your device manager and you should see the device renamed!



If you were successful, close out of the Zadig program andproceed to the next section!

Tiny AVR Programmer Hookup Guide - Programming in Arduino

If Zadig didn't work for you, check out the instructions below for help manually installing the drivers.

Manually Installing the libUSB Drivers

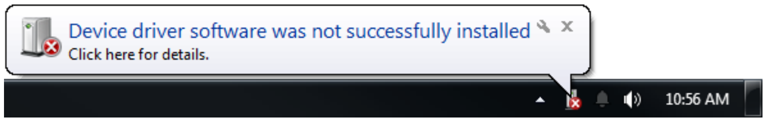
Step 1: Plug the Programmer In

To begin, locate an empty USB port on your computer, and plug the Tiny AVR Programmer into it. You'll probably want to have the programmer close by. If you're using a PC, or your USB ports aren't close by, a USB Extension Cable might help get the programmer into a more convenient spot on your desk.

Step 2: Wait for Windows to Automatically Fail/Succeed

After plugging in your Tiny AVR Programmer, Windows will try to look for a driver that matches it. Keep an eye on the notification area in the bottom-right corner.

Wait for Windows to try to install the driver on its own. There's a chance that, after searching, Windows will find the driver. If you get a *Device driver software installed successfully* notification (lucky you!), you can ignore the next few steps. But, if you got something like this:


Device driver software was not successfully installed
Click here for details.

Continue on to step 3...

Step 3: Download the Driver

If Windows couldn't find the driver for you, you'll need to download it. You can head over to the Tiny AVR Programmer GitHub repository and grab what you need there, or you can click the link below to download the zip file directly.

Tiny AVR Programmer Drivers (ZIP)

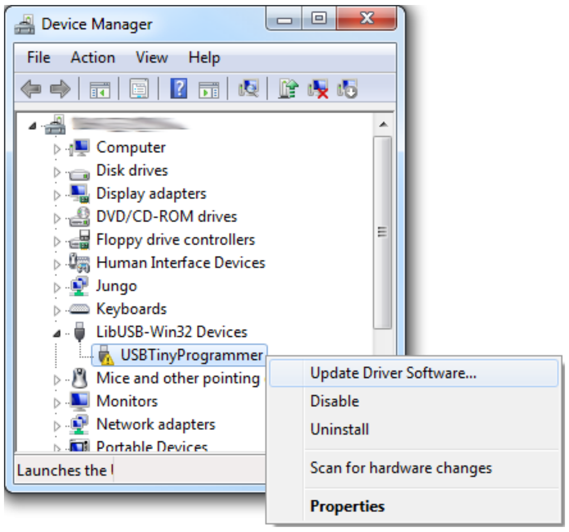
After downloading your driver, extract it from the zip folder. Don't forget where you put it!

Step 4: Open the Device Manager

To install the driver, you'll need to first open up the *Device Manager*. From the **Control Panel**, go to the **System and Security** section, click **System**, and click on **Device Manager**. (Alternatively you can **Run *devmgmt.msc***).

In the *Device Manager*, open up the *LibUSB-Win32 Devices* tree and you should find a *USBTinyProgrammer* with a yellow warning triangle hovering over the icon. This also may be located in "**Other devices** > **Unknown device**.

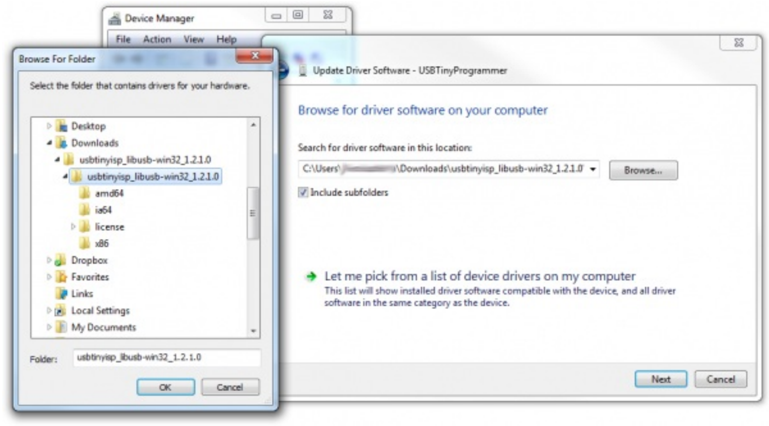
Right-click on the *USBTinyProgrammer* device, and select ***Update Driver Software...***



Step 5: Driver Pointing

On the *Update Driver Software* window that appears, **select *Browse my computer for driver software***.

On the next window, *Browse for driver software on your computer*, set the driver search location to the folder you downloaded and unzipped in step 3.



Select the folder that contains drivers for your hardware.

Folder: usbtinyisp_libusb-win32_1.2.1.0

Browse for driver software on your computer

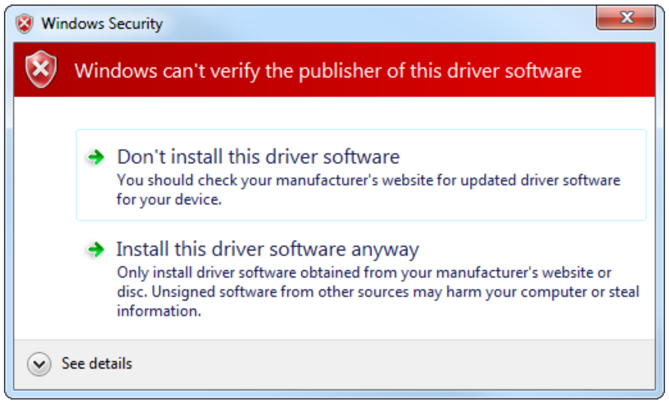
Search for driver software in this location:

C:\Users\[redacted]\Downloads\usbtinyisp_libusb-win32_1.2.1.0

[✓] Include subfolders

Let me pick from a list of device drivers on my computer
This list will show installed driver software compatible with the device, and all driver software in the same category as the device.

Then **click *Next***, and the driver will begin updating. Shortly after that, though, a *Windows Security* window should pop up to let you know the driver isn't "signed". **Click *Install this driver software anyway***. We promise it won't damage your computer!


Windows Security

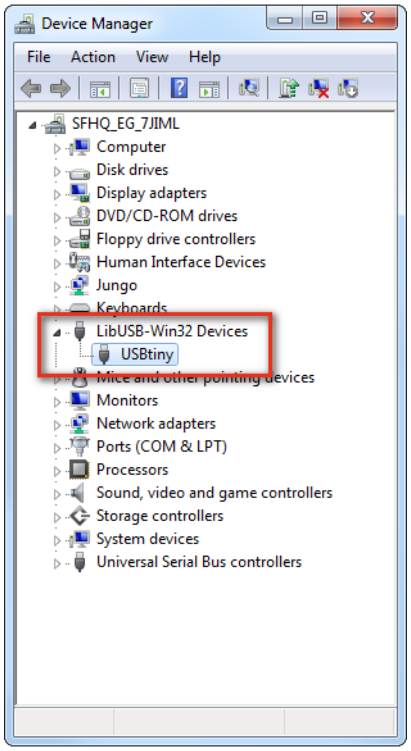
Windows can't verify the publisher of this driver software

Don't install this driver software
You should check your manufacturer's website for updated driver software for your device.

Install this driver software anyway
Only install driver software obtained from your manufacturer's website or disc. Unsigned software from other sources may harm your computer or steal information.

Then play the waiting game for a moment, and wait for a happy *Windows has successfully updated your driver software* window.

After closing that success window, your *Device Manager* should have an entry for ***USBtiny*** under *LibUSB-Win32 Devices*.



Congratulations! [Proceed over to the next section](), and we'll start using the Programmer!

Drivers Still Not Installing? If you are **still** having issues installing the drivers, try looking at this troubleshooting tip and driver from our technical support. The drivers work for both the Tiny AVR Programmer and Pocket AVR Programmer.

[GitHub SparkFunTechSupport: ...\PGM-11801]()

Breathe easy now! Once you've installed the USBtiny drivers on your computer, you shouldn't ever have to do it again. Now it's time to program something!

Programming in Arduino

Everyone loves Arduino! The simplified language makes programming AVRs and more complicated microcontrollers incredibly easy. Unfortunately, Arduino doesn't have any built-in functionality to program tiny AVRs, but that doesn't mean we can't add it!

On this page we'll go over all of the steps necessary to enable ATtiny45/85 programming in Arduino, using the Tiny AVR Programmer.

Step 0: Install Arduino

If you've never used Arduino before (where have you been?!), make sure you follow our [What is an Arduino?]() and [Installing Arduino]() tutorials before continuing on.

[What is an Arduino?]()

What is this 'Arduino' thing anyway? This tutorials dives into what an Arduino is and along with Arduino projects and widgets.
[Favorited Favorite]()47

Installing Arduino IDE

A step-by-step guide to installing and testing the Arduino software on Windows, Mac, and Linux.
Favorited Favorite 16

Step 1: Installing the ATtiny Add-On

The next step is to install the Attiny addon. The following steps in 1a and 1b will explain how to manually install the ATtiny board files for Arduino.

Tip: For beginners, you can **automatically** install using the Arduino boards manager by following the directions in "Installing the ATtiny Support in Arduino v1.6.4+."

High-Low Tech: Programming an ATtiny w/ Arduino 1.6 (or 1.0) - Installing ATtiny Support in Arduino 1.6.4

Step 1a: Download the ATtiny Addon

To manually add ATtiny's to the standard Arduino IDE *Board* menu, you'll need to add a few files that help define the hardware. The latest ATtiny hardware definitions are kept in a repository on GitHub.

GitHub ATtiny Boards

You can download them from there, or simply click on the archived links below (note: There are different files depending on which version of Arduino you are using):

- ATtiny for Arduino 1.0.x
- ATtiny for Arduino 1.6.x

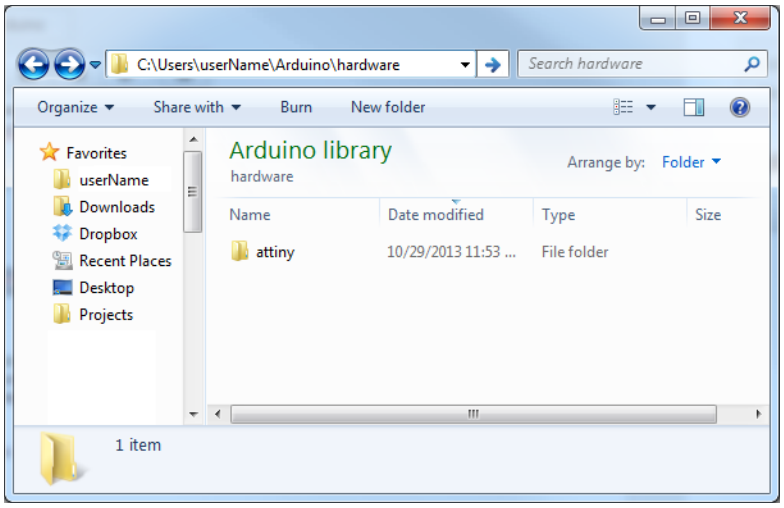
Extract the ZIP folder, and don't forget where you put it!

Step 1b: Move the *attiny* Folder

There should be an *attiny* folder living within the *attiny-ide-1.x.x.zip* file you downloaded. **Copy** that folder and **paste** it into a folder called ***hardware*** within your **Arduino Sketchbook directory**.

If you're not sure where your Arduino sketchbook is, **open Arduino** and go to ***File*** > ***Preferences***. The *Sketchbook location* should be the topmost entry in the *Preferences* dialog. By default, the sketchbook is usually an *Arduino* folder within your home folder (e.g. *C:\Users\userName\Arduino* on Windows, or */Users/userName/Documents/Arduino* on Mac).

If there's not a *hardware* folder already in your Sketchbook **make one**. After placing the *attiny* folder in there, your directory structure should look a little something like this:

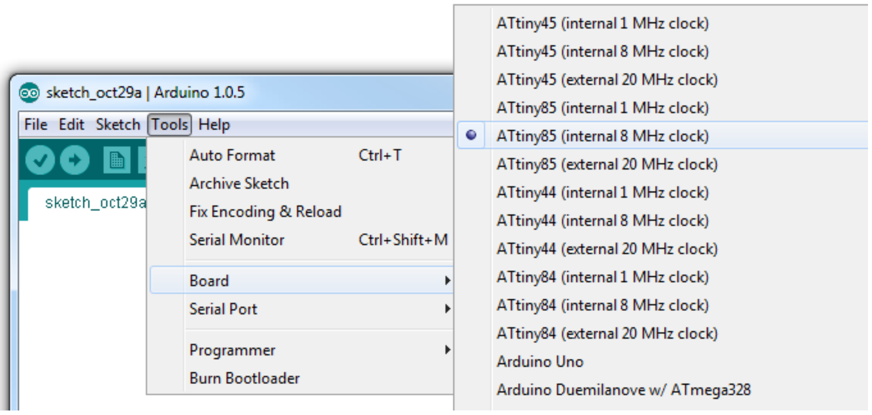


Step 2: Open and Configure Arduino

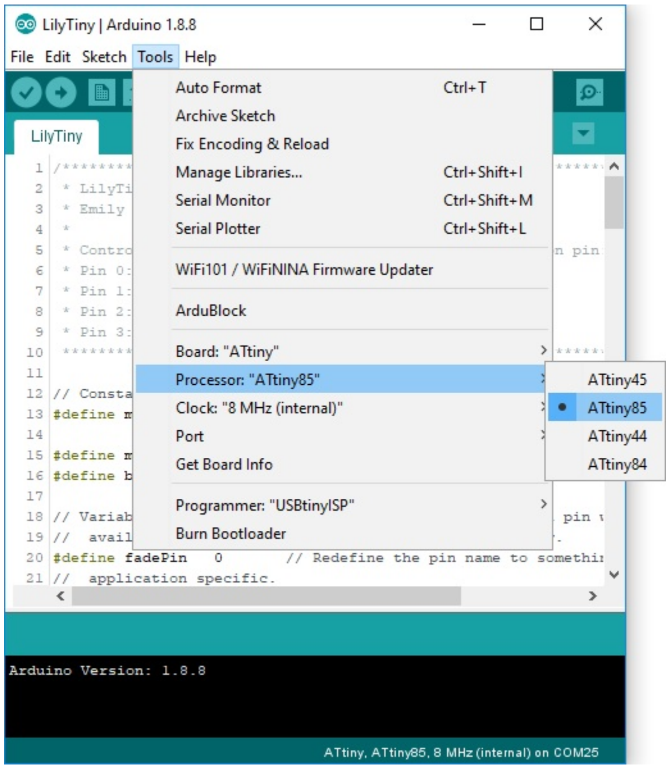
Almost to the fun part! **Open Arduino**. If you opened Arduino in the last step, close it and restart it.

Under the ***Tools*** > ***Board*** menu, you'll find the effects of the *attiny* folder. There should be twelve new entires in the board list, which allow you to program ATtiny45's, 85's, 44's and 84's. Each microcontroller can be set to a variety of clock speeds -- internal 1MHz or 8MHz or external 20MHz.

If you're using a bare, previously untouched ATtiny85 select ***ATtiny85 (internal 1 MHz clock)***. Be careful selecting here, selecting the *8 MHZ* option will only make your sketch run slow, but selecting the *20 MHz* option can "brick" your ATtiny. **Do not select the *20 MHz* option unless you have an external clock attached!**


sketch_oct29a | Arduino 1.0.5
File Edit Sketch Tools Help
sketch_oct29a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board
Serial Port
Programmer
Burn Bootloader
ATtiny45 (internal 1 MHz clock)
ATtiny45 (internal 8 MHz clock)
ATtiny45 (external 20 MHz clock)
ATtiny85 (internal 1 MHz clock)
ATtiny85 (internal 8 MHz clock)
ATtiny85 (external 20 MHz clock)
ATtiny44 (internal 1 MHz clock)
ATtiny44 (internal 8 MHz clock)
ATtiny44 (external 20 MHz clock)
ATtiny84 (internal 1 MHz clock)
ATtiny84 (internal 8 MHz clock)
ATtiny84 (external 20 MHz clock)
Arduino Uno
Arduino Duemilanove w/ ATmega328

Note: Depending on your Arduino IDE version, you may need to individually select the **attiny's Processor** (i.e. **ATtiny85**) and **Clock** (i.e. **8MHz (internal)**).

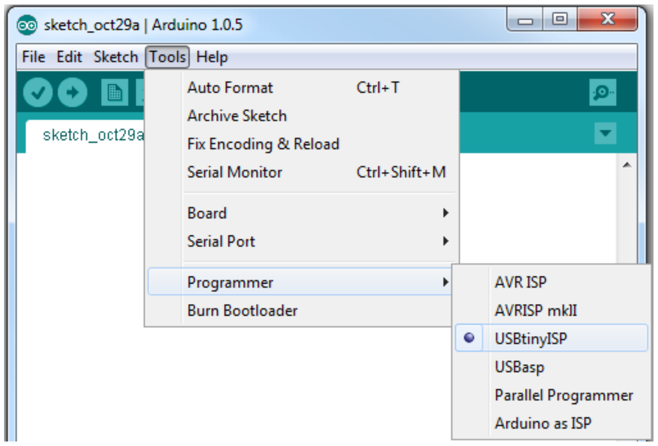


```
#define fadePin    0        // Redefine the pin name to somethi
//  application specific.
```

```
Arduino Version: 1.8.8
```

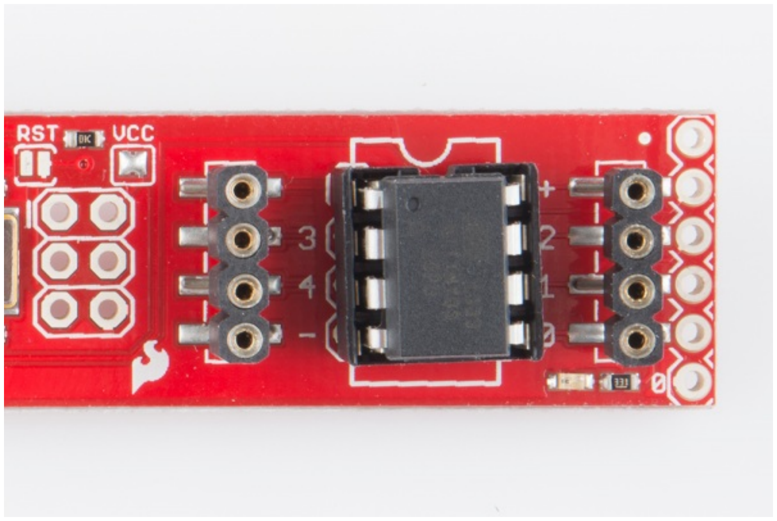
ATtiny, ATtiny85, 8 MHz (internal) on COM25

Unlike other Arduino boards, you don't have to select a *Serial Port* when using the Tiny AVR Programmer. But you do need to select a ***Programmer***. Under the *Tools* > *Programmer* menu, select ***USBtinyISP***.



Step 3: Plug in the ATtiny

Getting close to blinking! When you plug the ATtiny into your Programmer, make sure you get the polarity correct. The small, etched circle on the IC should line up with the "notch" on the Programmer's socket and silkscreen.



To get the IC into the socket, you may need to bend the legs on each side inwards a tad.

Step 4: Upload Code!

Time for the Blink sketch! The Tiny AVR Programmer has an on-board LED, connected to the ATtiny, which we can use to verify that code on the IC is running. The LED is connected to pin 0 in the Arduino environment. Copy/paste this code into your Arduino window:

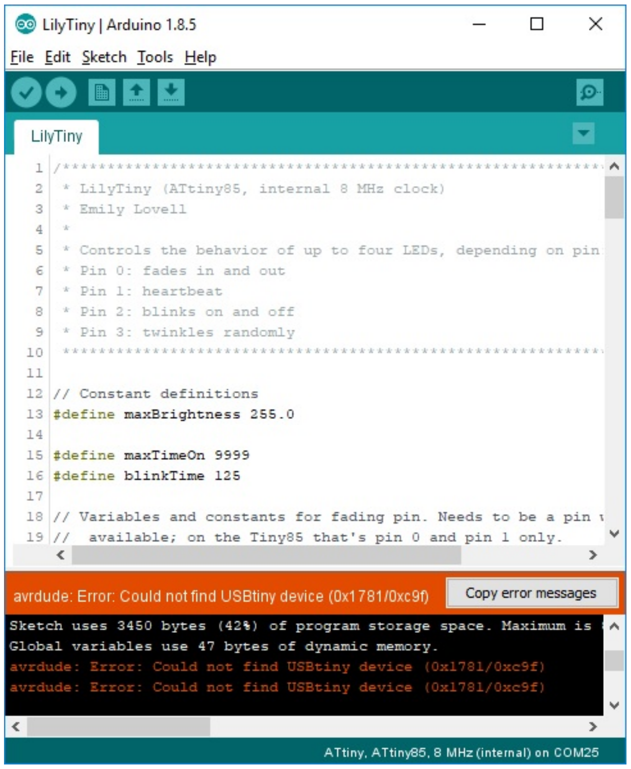
```
language:c
int blinkPin = 0;

void setup()
{
  pinMode(blinkPin, OUTPUT);
}

void loop()
{
  digitalWrite(blinkPin, HIGH);
  delay(500);
  digitalWrite(blinkPin, LOW);
  delay(500);
}
```

Then click the ***Upload*** button just as you would with any Arduino board. The code will compile, and then it should upload insanely fast. That's the wonders of direct in-system programming for you. If successful, the on-board **amber LED should start blinking**.

Note: You can also upload using the Arduino IDE menu. Depending on your Arduino IDE version, you can select eithe**Sketch** > **Upload Using Programmer** or **Sketch** > **Upload Using Programmer**.
Troubleshooting Tip: If you receive the following Arduino error, this might be due to the connection or an issue with the drivers.

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

You should see this at the bottom of the Arduino IDE in the text console.



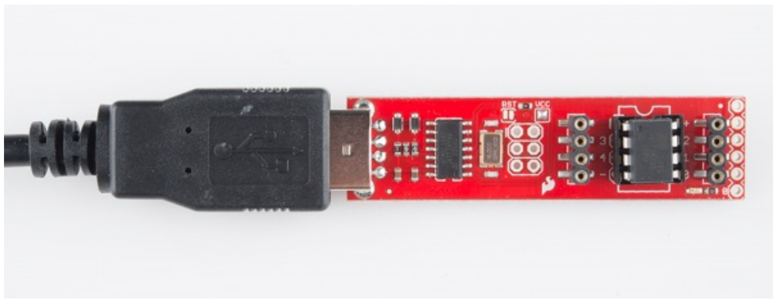
```
/*****************************************************************
 * LilyTiny (ATtiny85, internal 8 MHz clock)
 * Emily Lovell
 *
 * Controls the behavior of up to four LEDs, depending on pin
 * Pin 0: fades in and out
 * Pin 1: heartbeat
 * Pin 2: blinks on and off
 * Pin 3: twinkles randomly
 *****************************************************************

// Constant definitions
#define maxBrightness 255.0

#define maxTimeOn 9999
#define blinkTime 125

// Variables and constants for fading pin. Needs to be a pin w
//  available; on the Tiny85 that's pin 0 and pin 1 only.
```

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

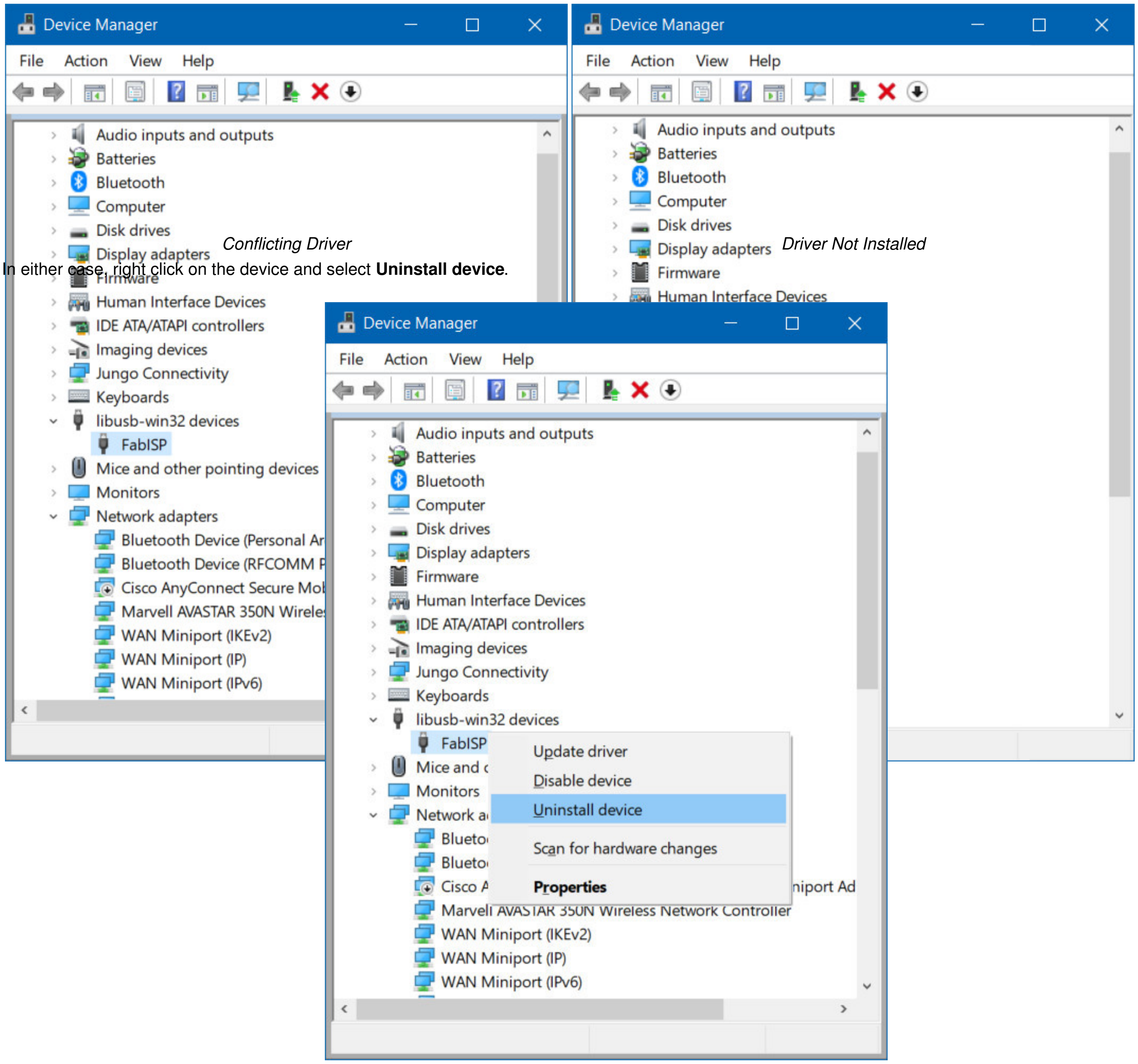
```
Sketch uses 3450 bytes (42%) of program storage space. Maximum is
Global variables use 47 bytes of dynamic memory.
avrdude: Error: Could not find USBtiny device (0x1781/0xc9f)
avrdude: Error: Could not find USBtiny device (0x1781/0xc9f)
```

ATtiny, ATtiny85, 8 MHz (internal) on COM25

If the issue is related to the connection, try unplugging/replugging the AVR programmer back into your USB cable or USB port. You also want to try a different USB cable.

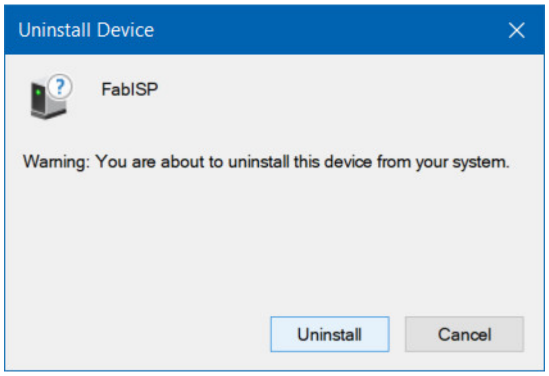


Otherwise, the issue may be due to the drivers. This may be caused by the driver not being installed correctly, or there is a driver conflict. Open up your device manager to view the device. The image on the left shows the device showing up as the **libusb-win32 devices** > **FabISP**. The image on the right shows the device showing up as **Other devices** > **FabISP**.



In either case, right click on the device and select **Uninstall device**.

You may see a window pop up similar to the image below. Click on the button labeled **Uninstall**. In some cases, Windows may provide an option to**"Delete the driver software for this device."** if the option is provided, simply mark the checkbox before clicking on the button to uninstall.



FabISP

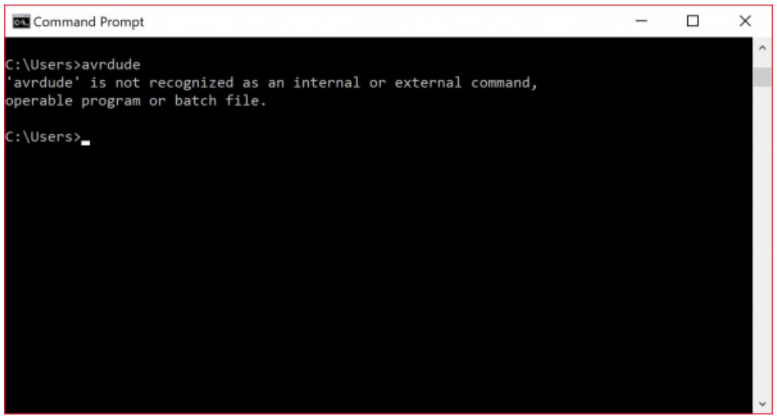
Warning: You are about to uninstall this device from your system.

Unplug and replug the AVR programmer back to your computer's USB port. Head back to the Driver Installation section and follow the instructions to Automatically Install the Drivers using Zadig.

Driver Installation: Automatically Install the Drivers with Zadig

Uploading Code the Hard Way

If you’re looking for more control over your Tiny AVR Programmer – and the AVR it’s connected to – follow along thetutorial for the Pocket AVR Programmer. While the tutorial was written for the Pocket AVR Programmer, it is functionally the same for the Tiny AVR Programmer. Just make sure to connect to the respective ICSP pins on the target AVR chip.



```
C:\Users>avrdude
'avrdude' is not recognized as an internal or external command,
operable program or batch file.

C:\Users>_
```

Using AVRDUDE via Command Line

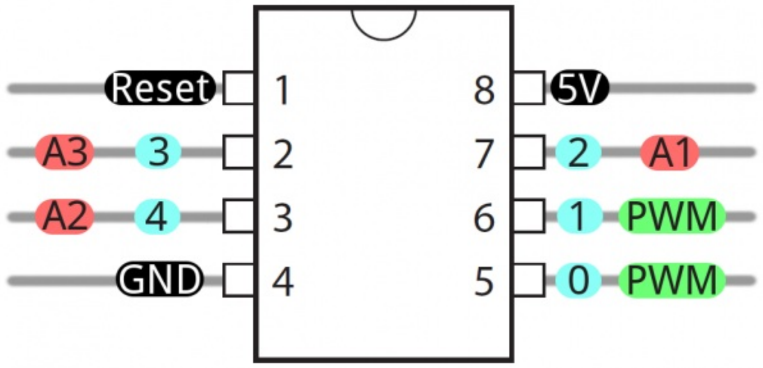
ATtiny85 Use Hints

The ATtiny85 isn't your everyday Arduino IC. It packs a lot of punch for its small size, but there are some things it can't do.

On this page, we'll provide a quick overview of the ATtiny85 as it pertains to Arduino and the Tiny AVR Programmer.

Pinout

Just like any Arduino board, each I/O pin on the ATtiny85 is assigned a numerical identifier. These pins are documented on the board as well, but you can also refer to the image below if you forget.



Each of the I/O pins on the ATtiny85 are capable of digital input and output. Beyond that, some pins have special functionality.

Analog Input and Output

There are **two analog outputs** and **three analog inputs**. Use them just as you would with any Arduino board. UseanalogWrite([pin], [0-255]) to do PWM output. This functionality is available on pins 0 and 1. For example:

```
language:c
int pwmPin = 0;

pinMode(pwmPin, OUTPUT);

for (int i=0; i<=255; i+=5)
{
  analogWrite(pwmPin, i);
  delay(5);
}
```

And use analogRead([pin]) to read an analog voltage between 0 and 5V, and turn it into a 10-bit representation of that voltage. Pins 2, 3, and 4 are capable of analog input, but, when using them as such, they should be referenced as **A1, A3, or A2** respectively. For example:

```
language:c
int pwmPin = 0;
int analogInPin = A1;

pinMode(pwmPin, OUTPUT);
pinMode(analogInPin, INPUT);

int analogIn = analogRead(analogInPin); // Read analog voltage on pin 2 (A1)
```

```
analogWrite(pwmPin, analogIn / 4); // Output analog reading to dimmable LED
```

Note: For advanced users, you can modify the ATtiny85's timer/counter registers to increase the number of PWM channels available! You can get up to four PWM channels by following this tutorial here: Technoblogy - Four PWM Outputs from the ATtiny85. The example just runs PWM on the ATtiny85. If you are trying to add more features outside of the PWM example, there might be some unexpected behaviors since it is modifying the timing registers.

If you need to reset the chip, simply use the erase command with the Tiny AVR or Pocket AVR Programmer via command line to get it back to its previous state. Uploading code with the Arduino IDE will not be enough. Here is an example using the fuse bit settings for the LilyTwinkle's ATtiny85:

```
avrdude -c usbtiny -b 19200 -p t85 -v -e -U lfuse:w:0xe2:m -U hfuse:w:0xdf:m -U efuse:w:0xff:m -U lock:w:0xCF:m
```

After resetting the chip, you can proceed to upload code to the chip through the Arduino IDE as explained earlier.

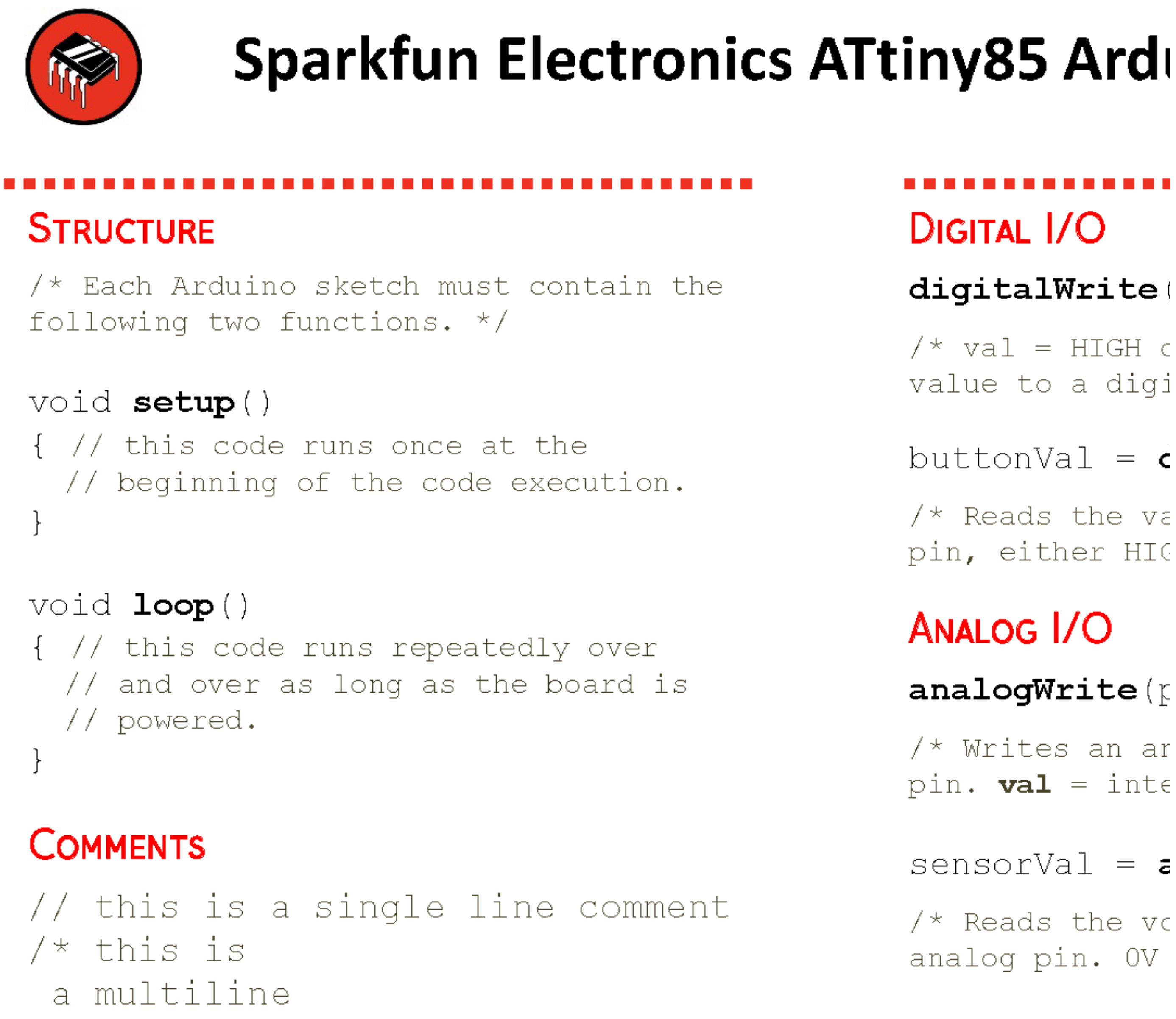
No Serial (UART). Yes SPI and I^{2}C.

You may notice, on the listing of special pin functions there are no UART RX's or TX's. That's because the ATtiny85 doesn't have a built ihardware UART. If you try to compile any Arduino code with `Serial.begin(9600)`'s or `Serial.print()`'s you'll get an error.

So you're out one of the more useful Arduino debugging tools. You can't print to the Serial Monitor. But the ATtiny85 does still have I^{2}C and SPI, which are much more commonly used for sensor communication these days. Unfortunately, the Arduino libraries for these interfaces haven't yet been written for the ATtiny85, but there are some user contributed libraries around the web. USIi2c is an Arduino library which enables I^{2}C on the ATtiny85.

There are other ATtiny85-focused libraries out there too. Like aServo8Bit, a servo library.

Tip: Looking for a quick reference guide for the ATtiny85? Click on the link below to download an image or PDF version from our resources!



Sparkfun Electronics ATtiny85 Ard

STRUCTURE

```
/* Each Arduino sketch must contain the
following two functions. */

void setup()
{ // this code runs once at the
  // beginning of the code execution.
}

void loop()
{ // this code runs repeatedly over
  // and over as long as the board is
  // powered.
}
```

COMMENTS

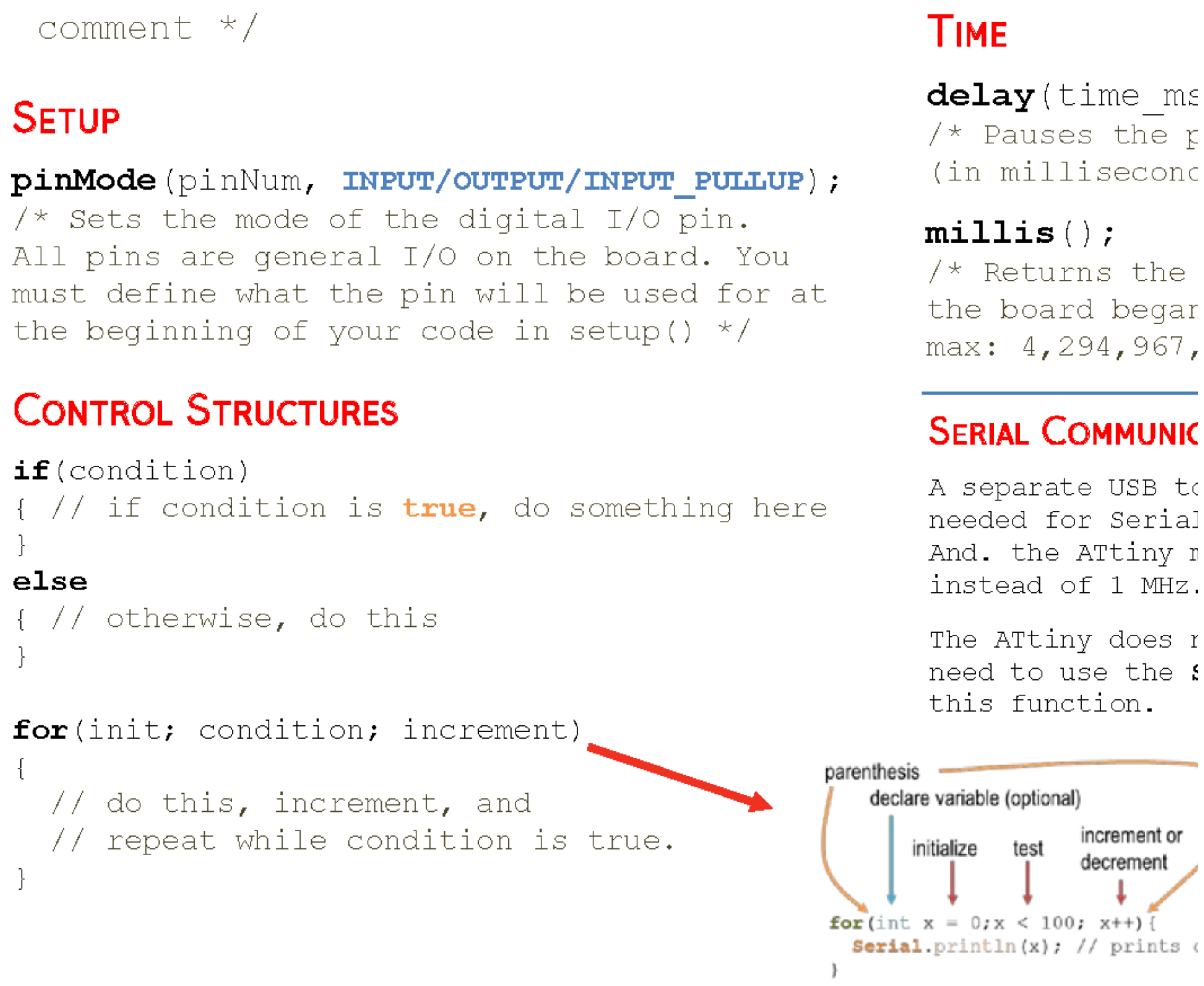
```
// this is a single line comment
/* this is
 a multiline
```

DIGITAL I/O

```
digitalWrite(

/* val = HIGH o
value to a digi

buttonVal = d

/* Reads the va
pin, either HIG
```

ANALOG I/O

```
analogWrite(p

/* Writes an ar
pin. val = inte

sensorVal = a

/* Reads the vo
analog pin. 0V
```



```
comment */
```

SETUP

```
pinMode(pinNum, INPUT/OUTPUT/INPUT_PULLUP);
/* Sets the mode of the digital I/O pin.
All pins are general I/O on the board. You
must define what the pin will be used for at
the beginning of your code in setup() */
```

CONTROL STRUCTURES

```
if(condition)
{ // if condition is true, do something here
}
else
{ // otherwise, do this
}

for(init; condition; increment)
{
  // do this, increment, and
  // repeat while condition is true.
}
```

TIME

```
delay(time_ms
/* Pauses the p
(in milliseconc

millis();
/* Returns the
the board begar
max: 4,294,967,
```

SERIAL COMMUNIC

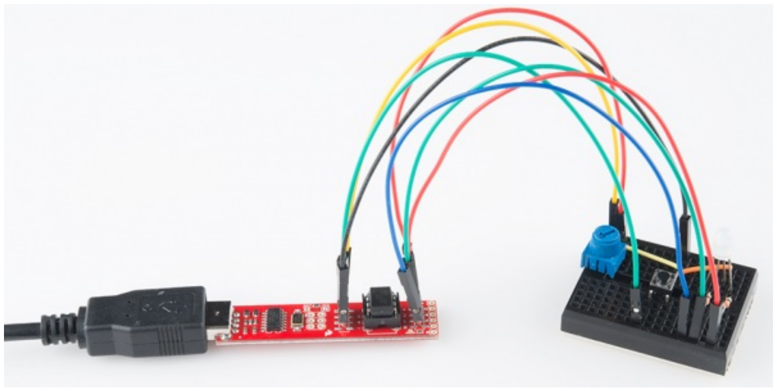
```
A separate USB tc
needed for Serial
And. the ATtiny r
instead of 1 MHz.

The ATtiny does r
need to use the S
this function.
```

```
for(int x = 0;x < 100; x++){
  Serial.println(x); // prints 
}
```

Prototyping with the Tiny AVR Programmer

There's only so much excitement you can get out of dimming a single, yellow LED. You'll eventually want to branch out, and start connecting your tiny85 to other electronic components. There are a few ways to do this.

The easiest, least permanent prototyping route is to use the**prototyping headers** on either side of the socket. You can connect standard,male jumper wires to these pins, which can in turn be routed to breadboards or other components.



For more permanent projects, it's easy enough to gently**remove the IC from the socket**, and plug it into a PCB or breadboard. Eventually, once you've iterated enough on your sketch, this is probably where you'll want to go. Eventually you arrive at finished designs like the H2OhNo! or the LectroCandle.

Surface Mount ATtiny85 SOIC Packages

Trying to reprogram an ATtiny85 with a SOIC Package? There are a few ways to connect. The easiest would be to use the IC test clip and M/F jumper wires.

[Jumper Wires Premium 6" M/F Pack of 10]

PRT-09140
$4.50
1
Favorited Favorite 16
Wish List

IC Test Clip - SOIC 8-Pin

COM-13153
$19.50
4
Favorited Favorite 24
Wish List

For more information, check out our tutorial on reprogramming ATtiny85's on the LilyTiny and LilyTwinkle.

Re-Programming the LilyTiny / LilyTwinkle

September 11, 2014

A quick tutorial showing how to reprogram the ATtiny85 IC found on the LilyTiny or LilyTwinkle boards.
Favorited Favorite 4

If the programming pins are broken out on a standard 2x3 ICSP header, you could also solder together the ISP pogo adapter to temporarily connect to the chip. Or you could grab a few alligator test leads or IC hook to individually connect each programming pin to the Tiny AVR Programmer's machine headers.

Alligator Clip with Pigtail (10 Pack)

CAB-14303
$7.50
4
Favorited Favorite 13
Wish List

IC Hook with Pigtail

CAB-09741
$5.50
10
Favorited Favorite 29

Wish List

SparkFun ISP Pogo Adapter

KIT-11591
4
Favorited Favorite 12
Wish List

Resources and Going Further

Now that you've successfully got your Tiny AVR Programmer up and running, it's time to incorporate it into your own project!

For more information, check out the resources below:

- **Tiny AVR Programmer Design Files**
  - Schematic -- A PDF of the Tiny AVR Programmer's schematic.
  - Eagle Files -- If you want to look at the PCB design, or modify it to make a version of your own, check these files out.
  - Tiny AVR Programmer Firmware -- If you want to dig into the code that lives on the Programmer itself, check this out.
  - GitHub Repository -- Go here to find the latest, greatest version of the Tiny AVR Programmer's hardware and firmware. Or modify it and contribute your changes back!
- **Drivers, etc.**
  - Arduino Board Definitions -- The *attiny* folder should live within a *hardware* folder in your Arduino sketchbook.
  - Zadig v2.0.1.160 Software and USBtiny (ZIP) -- For automatic installation
  - 32-bit USBTinyISP Driver -- Windows driver for manual installation on 32-bit systems
  - 64-bit USBTinyISP Driver -- Windows driver for manual installation on 64-bit systems
  - GitHub: Signed USBTiny Drivers -- If the first 3 options listed in the tutorial fail to install fail to install
- **ATtiny85 Resources**
  - Atmel Documentation Page -- The latest datasheets and application notes for the ATtiny85.
  - High-Low Tech Tutorial -- An overview of programming the ATtiny85 using Arduino.
  - Quick Reference Guide for ATTiny85

Need some inspiration for your next project? Check out some of these related tutorials:

- H2OhNo! -- The H2OhNo! water alarm and development board uses an ATtiny85 to sense the presence of water. This tutorial goes deep into getting the ATtiny85 into a very **low power** mode.
- Shift Registers -- If you're feeling restrained by the ATtiny's lack of pins, you may be able to use a shift register to expand on that I/O count.
- Using the Arduino Pro Mini 3.3V -- If you're looking for small, but need more pins and functionality check out the Arduino Pro Mini.
- Installing an Arduino Bootloader -- You can use the Tiny AVR Programmer to program all sorts of AVRs, including those on most Arduino-compatible boards. If you ever find yourself needing to reprogram your Arduino bootloader, the Tiny AVR Programmer should be all you need.
- Pocket AVR Programmer Hookup Guide -- If you are interested in using the Tiny AVR Programmer to program other AVRs with AVRDUDE via command line, this tutorial will provide a few examples and troubleshooting tips. Just make sure to connect to the respective ICSP pins.

Installing an Arduino Bootloader

This tutorial will teach you what a bootloader is and why you would need to install or reinstall it. We will also go over the process of burning a bootloader by flashing a hex file to an Arduino microcontroller.
Favorited Favorite 25

Shift Registers

An introduction to shift registers and potential uses.
Favorited Favorite 44

Using the Arduino Pro Mini 3.3V

This tutorial is your guide to all things Arduino Pro Mini. It explains what it is, what it's not, and how to get started using it.
Favorited Favorite 16

Pocket AVR Programmer Hookup Guide

Skip the bootloader and load your program directly onto an AVR with the AVR Pocket Programmer.
Favorited Favorite 10

Or check out these tutorials using the Attiny.

The Uncertain 7-Cube

The Uncertain 7-Cube is a non-committal, less-than-helpful, but also entirely honest fortune teller. Simply ask it a yes or no question, give it a nudge, and the 7-Cube will dutifully inform you that it doesn't have all the facts and doesn't feel comfortable making a guess.
Favorited Favorite 3

Re-Programming the LilyTiny / LilyTwinkle

A quick tutorial showing how to reprogram the ATtiny85 IC found on the LilyTiny or LilyTwinkle boards.
Favorited Favorite 4

Electronic E-craft Terrarium

A guest tutorial on an Electronic Terrarium from the Performative Sculpture class of Parsons DT.
Favorited Favorite 3

Twinkle Zodiac Constellation

Create a quick and easy piece of e-textile art based on a Zodiac sign using a LilyTiny, LilyPad LEDs, battery holder, conductive thread, and coin cell battery.
Favorited Favorite 12

Are you looking to use a Pi to flash larger file sizes to your AVR microcontrollers? Try checking out the Pi AVR Programmer Hat!

Pi AVR Programmer HAT Hookup Guide

July 26, 2018

In this tutorial, we will use a Raspberry Pi 3 and the Pi AVR Programmer HAT to program an ATMega328P target. We are going to first program the Arduino bootloader over SPI, and then upload an Arduino sketch over a USB serial COM port.
Favorited Favorite 2