

ATtiny85 Light Show



CYBER RESILIENT ENERGY
DELIVERY CONSORTIUM

Explore programming and circuits by using a micro controller to make a creative story! A micro controller can take simple instructions and execute them repeatedly. You can program multiple LEDs to create a light show using your knowledge of computer science and other simple tools. This tutorial is for windows computers.

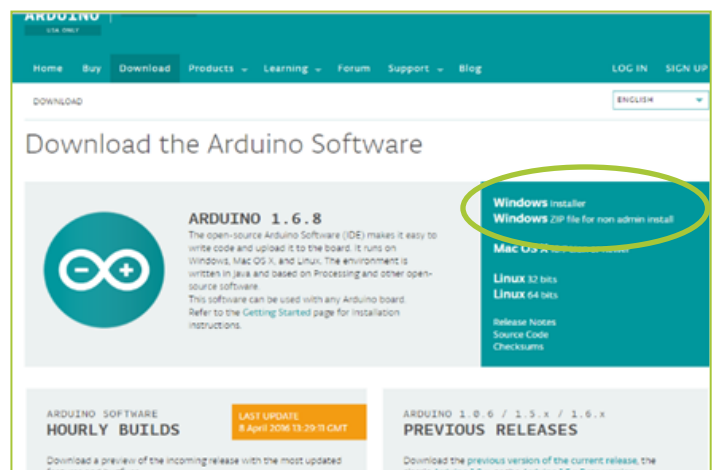
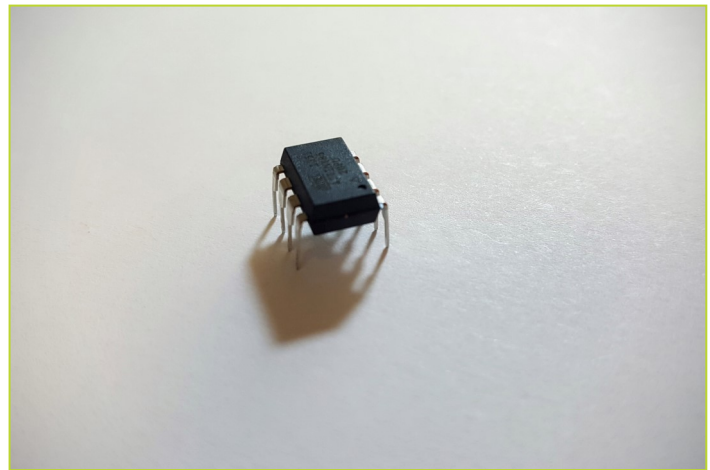
Note: You will need to download the Arduino software onto your computer in order to program your ATtiny!

MATERIALS

- LEDs
- Copper Tape
- Tiny AVR Programmer
- ATtiny85 or Other
- Computer
- Scissors
- Coin Battery
- Nonconductive Tape

Download Arduino IDE

1. Go to www.arduino.cc → Download → Select the windows installer.
2. Go to the location where your downloads are stored and run the installer. The name should be something along the line of “Arduino-1.6.8-windows” where the number is the current version of the program.
 - a. Agree to the License Agreement
 - b. Install everything on the list, Next >
 - c. Choose your desired installation location. ‘Install’
 - d. Wait for the installation to finish, when complete, ‘close’



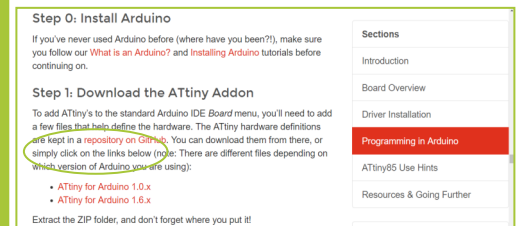
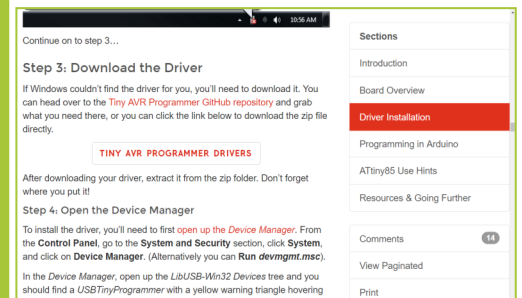
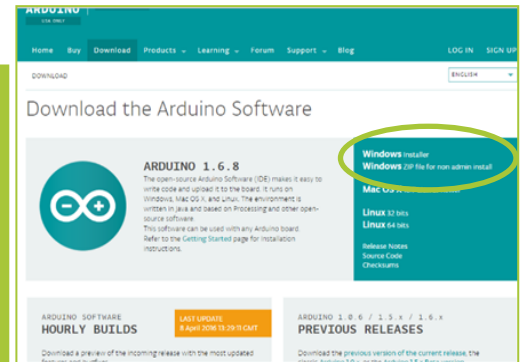
The Cyber Resilient Energy Delivery Consortium (CREDC) Education team continues the work of the TCIPG Education project. The team develops interactive lessons and activities designed to link researchers, educators, consumers, and students. The materials illustrate challenges, trade-offs, and decisions required for secure and economical power delivery. The project seeks to involve families learning together while creating interest in STEM disciplines and careers. The project website offers a variety of hands-on and virtual energy related activities and challenges. credc.mste.illinois.edu

Installing the Plugins and Drivers

Downloading the plugins and drivers

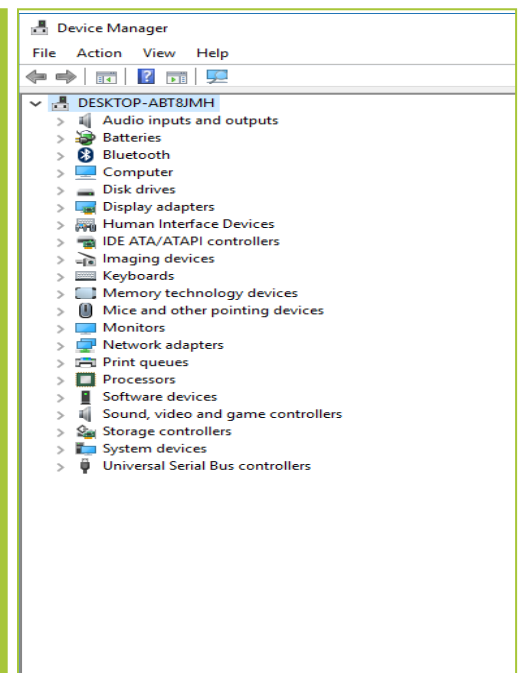
1. Insert the AVR programmer into an accessible USB port into your computer.
2. Open up your web browser and go to www.sparkfun.com. Click on 'TUTORIALS'. Click in the search box and search "Tiny AVR Programmer Hookup Guide" and click on the result with that title.
3. In "Sections" on the right side of the screen, click "Driver Installation" and then find the big button that says "TINY AVR PROGRAMMER DRIVERS". Click on it to download a ZIP folder with the required drivers for the Programmer. After you have downloaded the drivers, scroll down the page to the "Programming in Arduino" section, and download the ZIP folder "ATtiny for Arduino 1.6.x".

Pay attention to where your files go!



Installing the plugins and drivers

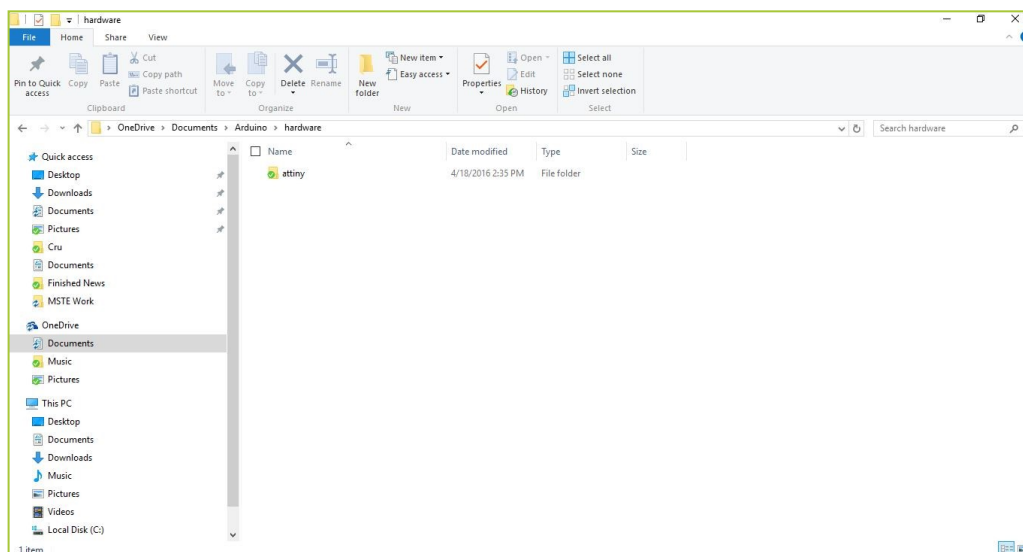
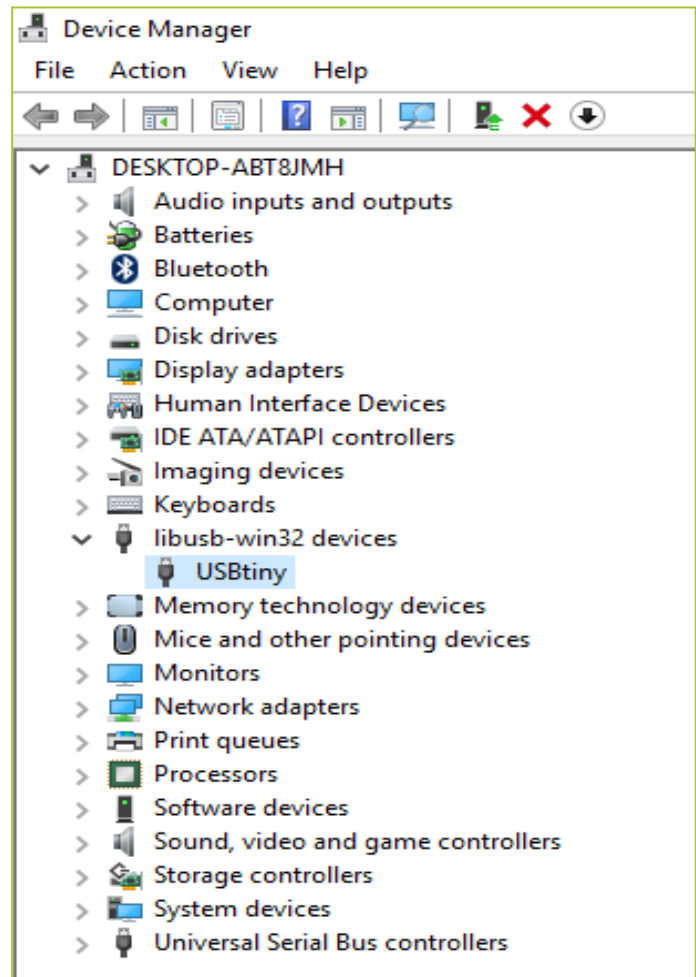
1. Extract the ZIP folders you just downloaded to a place you can remember.
2. Open the device manager for your computer. Open the device manager list. Go to Control Panel → System and Security → System → Device Manager .
3. Look for where the Tiny AVR programmer appears on the list. It should appear as "Other Devices" → "FabSP" or "LibUSB-Win32 Devices" → "USBTinyProgrammer"



Installing the Plugins and Drivers continued

Installing the plugins and drivers continued

- Right click on the title of the USBTinyProgrammer, and click on “Update Driver Software”. Manually browse to find the folder you unzipped with the driver software. Use this as the new updated drivers. Click next, and the drivers will begin updating. You’ll probably get a windows security popup asking you if you’re sure you want to install. Tell it to install, and wait until it’s finished. After it has completed, you have successfully updated the drivers for the AVR Tiny Programmer! Congratulations!
- Open the Arduino IDE that you downloaded in step 1. Go to file > preferences, and go to the directory that is shown in “sketchbook location:” Now, drag the “ATtiny” folder that you extracted from the other ZIP folder into the “hardware” folder in the sketchbook. If there is no hardware folder create one now and copy the folder in.



Using the AVR Programmer with Arduino

Options for Arduino IDE

If you haven't yet, close the IDE and reopen it. Next, go to "tools" then change the "board", "processor", and "clock" to the following settings:

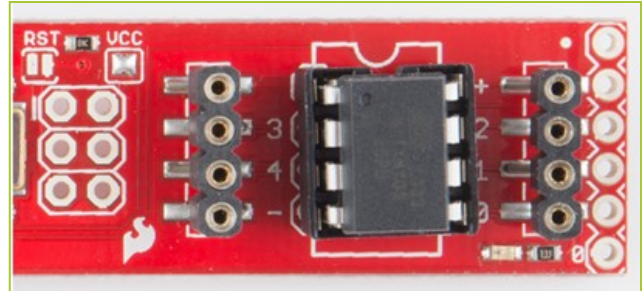
Board: "ATtiny"

Processor: "ATtiny85"

Clock: "1 MHz (internal)"

Programmer: "USBTinyISP"

(It is important to choose 1MHz because you can break your ATtiny85 otherwise.)



How to use the ATtiny85 microcontroller

When plugging in the microcontroller into the AVR programmer, use caution so you do not break any of the legs on the ATtiny. On the picture above, notice that the dot on the top right corner of the microcontroller is in the same direction as the notch on the white outline. You will have to bend the legs a little bit to get the ATtiny to fit well in the slot.

```
sketch_may02a | Arduino 1.6.8
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
Board: "ATtiny" >
Processor: "ATtiny85" >
Clock: "1 MHz (internal)" >
Port >
Programmer: "USBTinyISP" >
Burn Bootloader

void setup()
{
  pinMode(0,
  pinMode(2,
}

void loop()
{
  // digitalWrite(2, HIGH);
  // delay(500);
  // digitalWrite(2, LOW);
  // delay(500);
  // digitalWrite(0, HIGH);
  // delay(1000);
  for (int i = 0; i < 5; i++) {
    strobe0();
    for (int j = 0; j < 2; j++) {
      strobe2();
    }
    strobe0();
  }
}
```

ATtiny, ATtiny85, 1 MHz (internal) on COM1

Congratulations!

You are now ready to begin programming your ATtiny85 for your story! Read on to learn more about programming in the Arduino IDE.

Programming in Arduino

Example Code

To the left, you'll see the code I wrote for my story! Anytime you see some word or name followed by "(" (i.e. name()) and then curly braces with code in the middle (i.e. { . . . }) you have a "method". The "setup()" and "loop()" methods are looked at by the compiler as the actual code that will be executed. As you can see, I have two lines of code in the "setup()" method, pinMode(0, OUTPUT); pinMode(2, OUTPUT);. You will need this in your code too! It tells the microcontroller that pin 0 on the microcontroller will be set to outputting electricity rather than input or sensing electricity.

Inside the "loop()" method you will write the code that controls your light! You can see the important statements in the "onOff0()" and "onOff2()" method at the bottom:

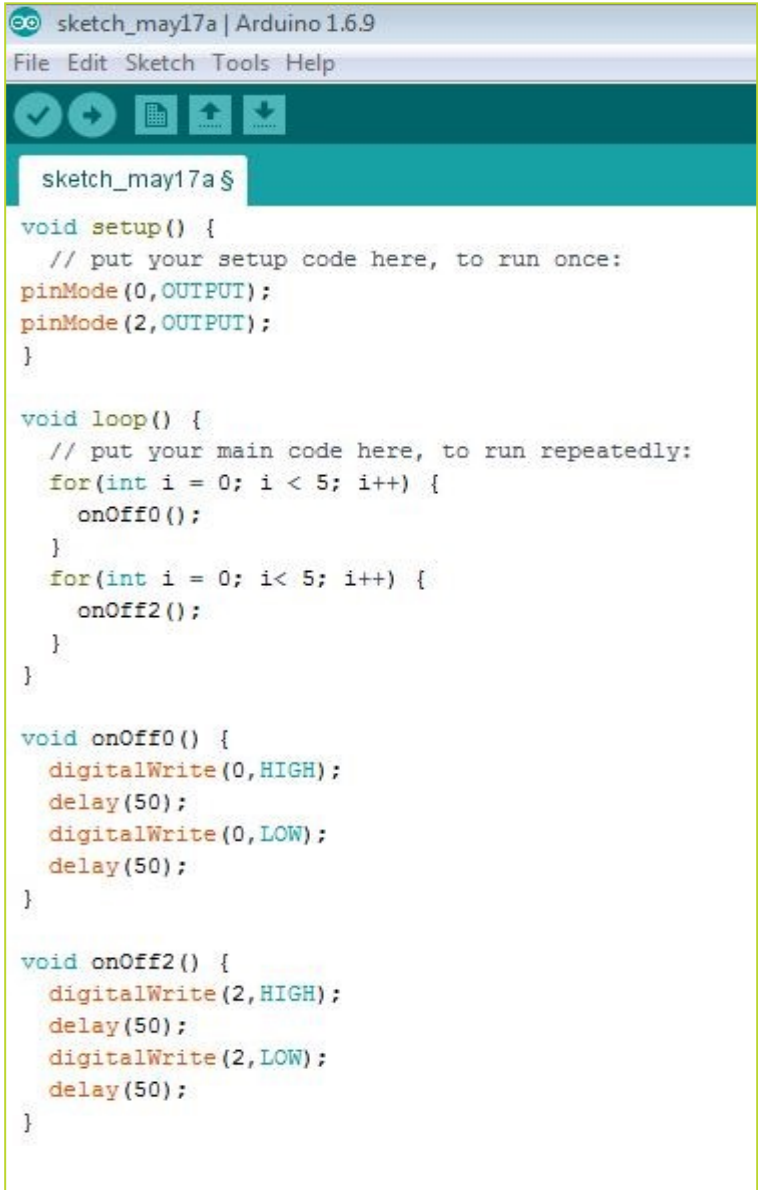
digitalWrite(0, HIGH) : sends electricity through pin 0, which turns on the light!

delay(anyinteger) : do nothing for the number of milliseconds inside the delay.

digitalWrite(0, LOW) : Stops sending electricity through pin 0.

In my code, I have 2 'for' loops. This loop runs the code within the { } a specified amount of times. In my case, 5 each. The 'for' loops run methods called "onOff0()" and "onOff2()" which contain the important part of my code. You can see why I chose a 'for' loop, since I wanted to run the code within onOff0() and onOff2() 10 times collectively. Since onOff0() and onOff2() each contain 4 lines of code, I would have had to write 40 lines to accomplish the same goal. My code will look like this:

1. Run onOff0() 5 times within a 'for' loop.
2. Run onOff2() 5 within a 'for' loop
3. onOff0()and onOff2() is a strobe light effect, turning on the light for 50 ms and turning it off for 50 ms.
4. Return to 1.



```
sketch_may17a | Arduino 1.6.9
File Edit Sketch Tools Help

sketch_may17a $

void setup() {
  // put your setup code here, to run once:
  pinMode(0, OUTPUT);
  pinMode(2, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  for(int i = 0; i < 5; i++) {
    onOff0();
  }
  for(int i = 0; i < 5; i++) {
    onOff2();
  }
}

void onOff0() {
  digitalWrite(0, HIGH);
  delay(50);
  digitalWrite(0, LOW);
  delay(50);
}

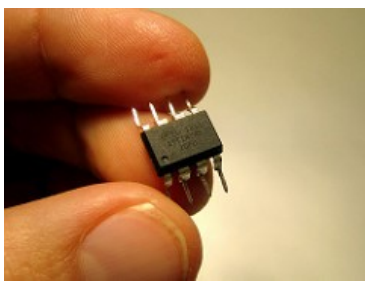
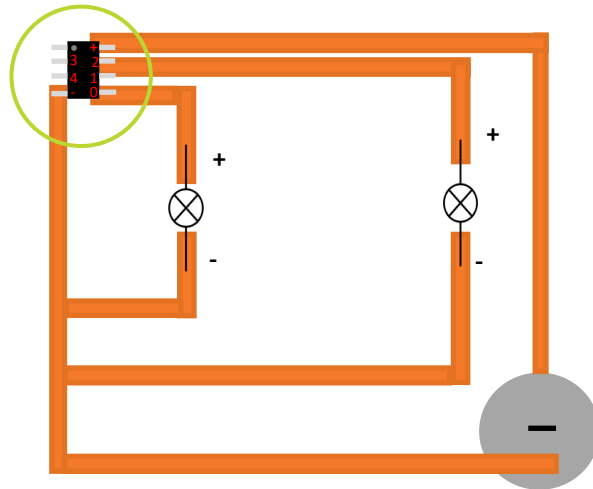
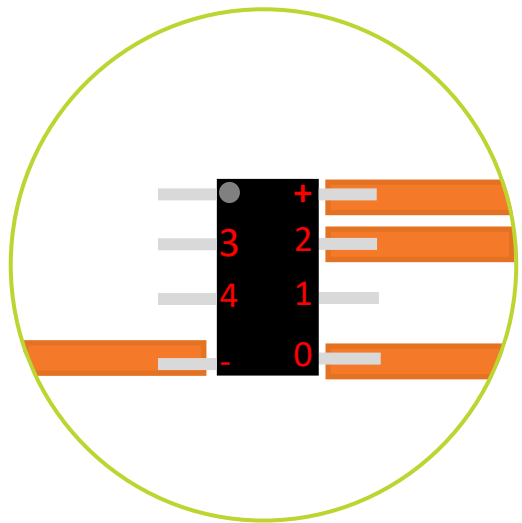
void onOff2() {
  digitalWrite(2, HIGH);
  delay(50);
  digitalWrite(2, LOW);
  delay(50);
}
```

Check out the "Programming in Arduino" PDF for more in depth instructions!

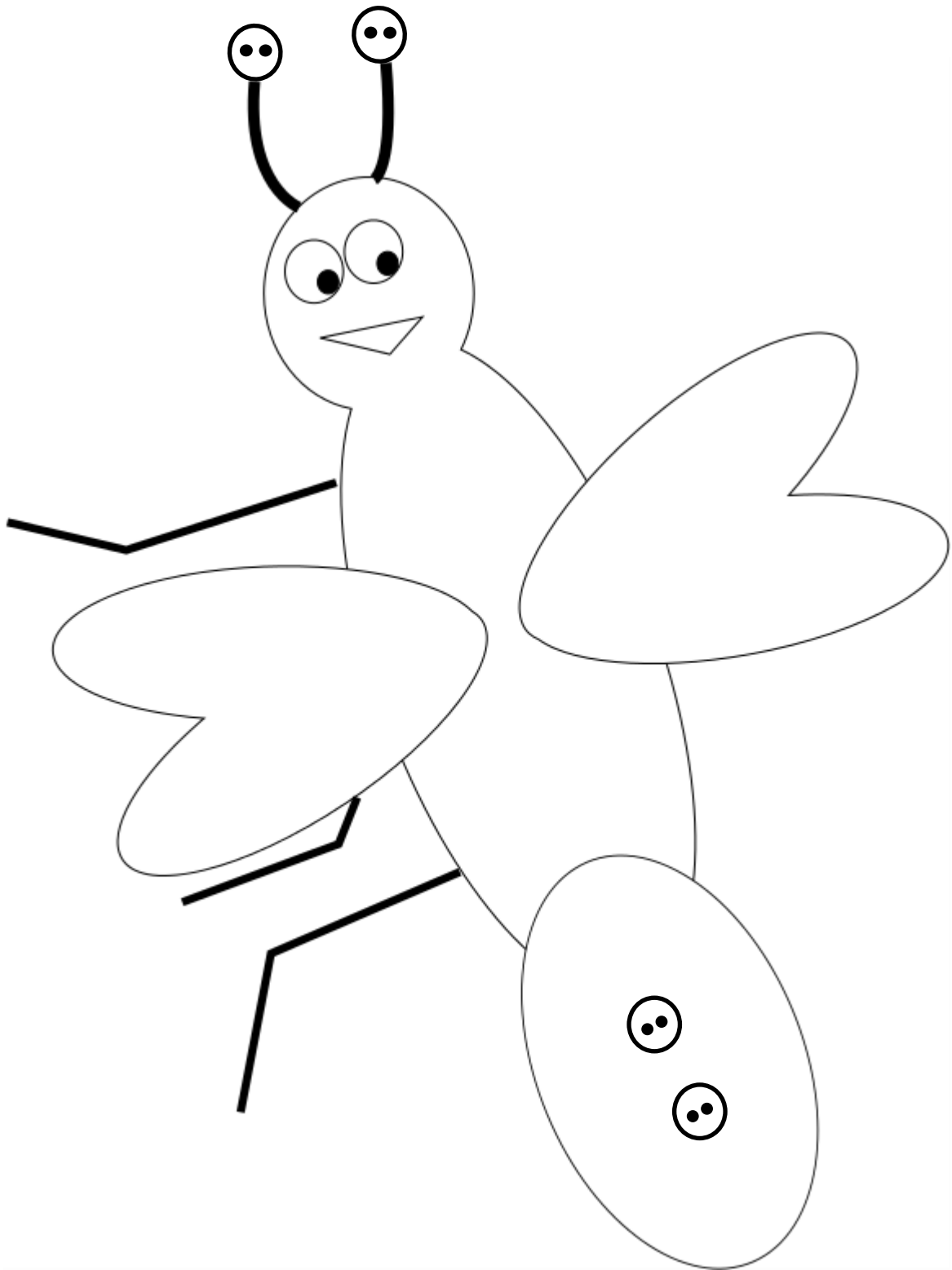
Programming in Arduino

Decorate your house!





(Fold the legs of the ATtiny85 like this)



Key

LED



Tape Here



ATTiny85 (grey circle is the notch on the corner)



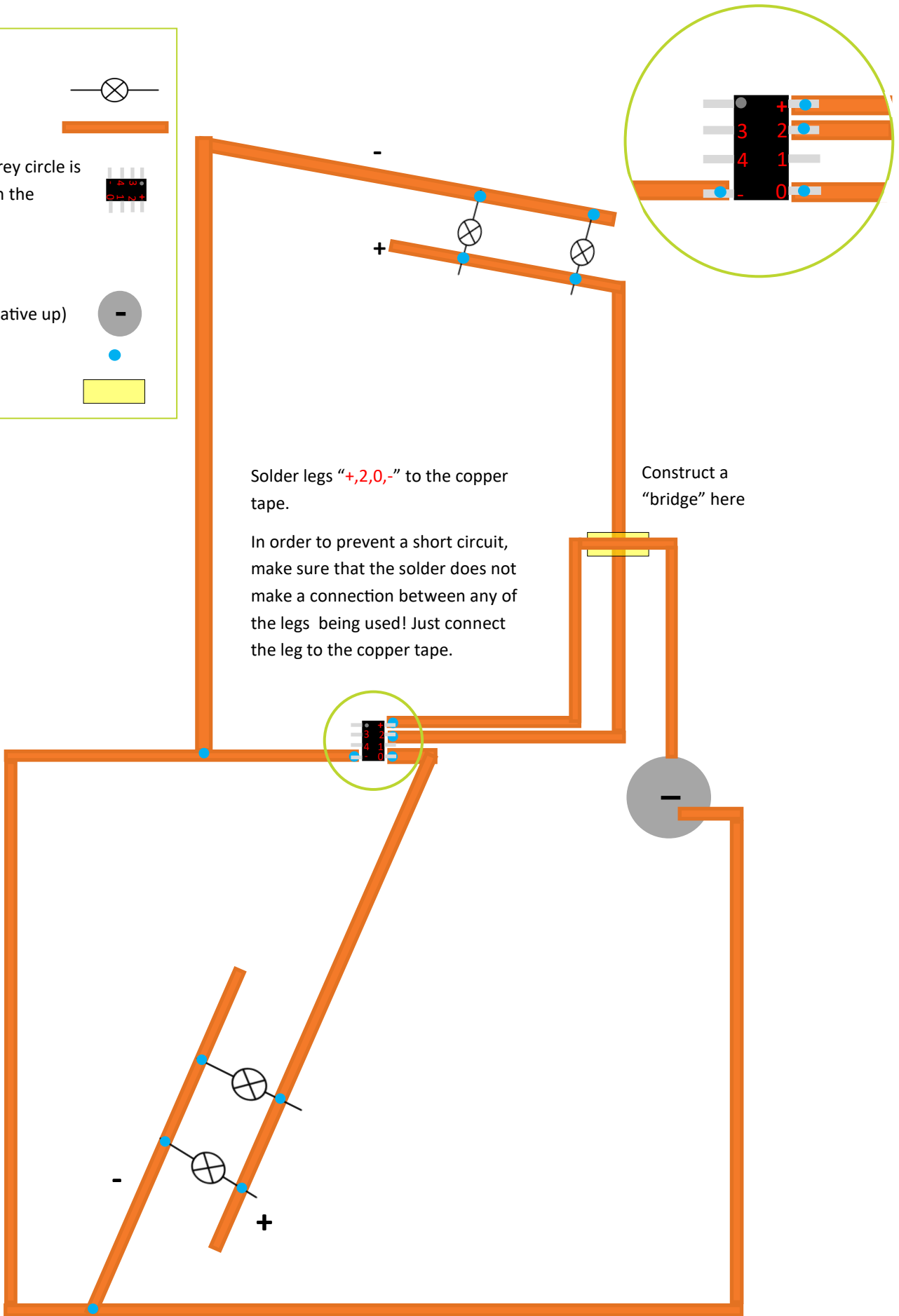
Battery (negative up)

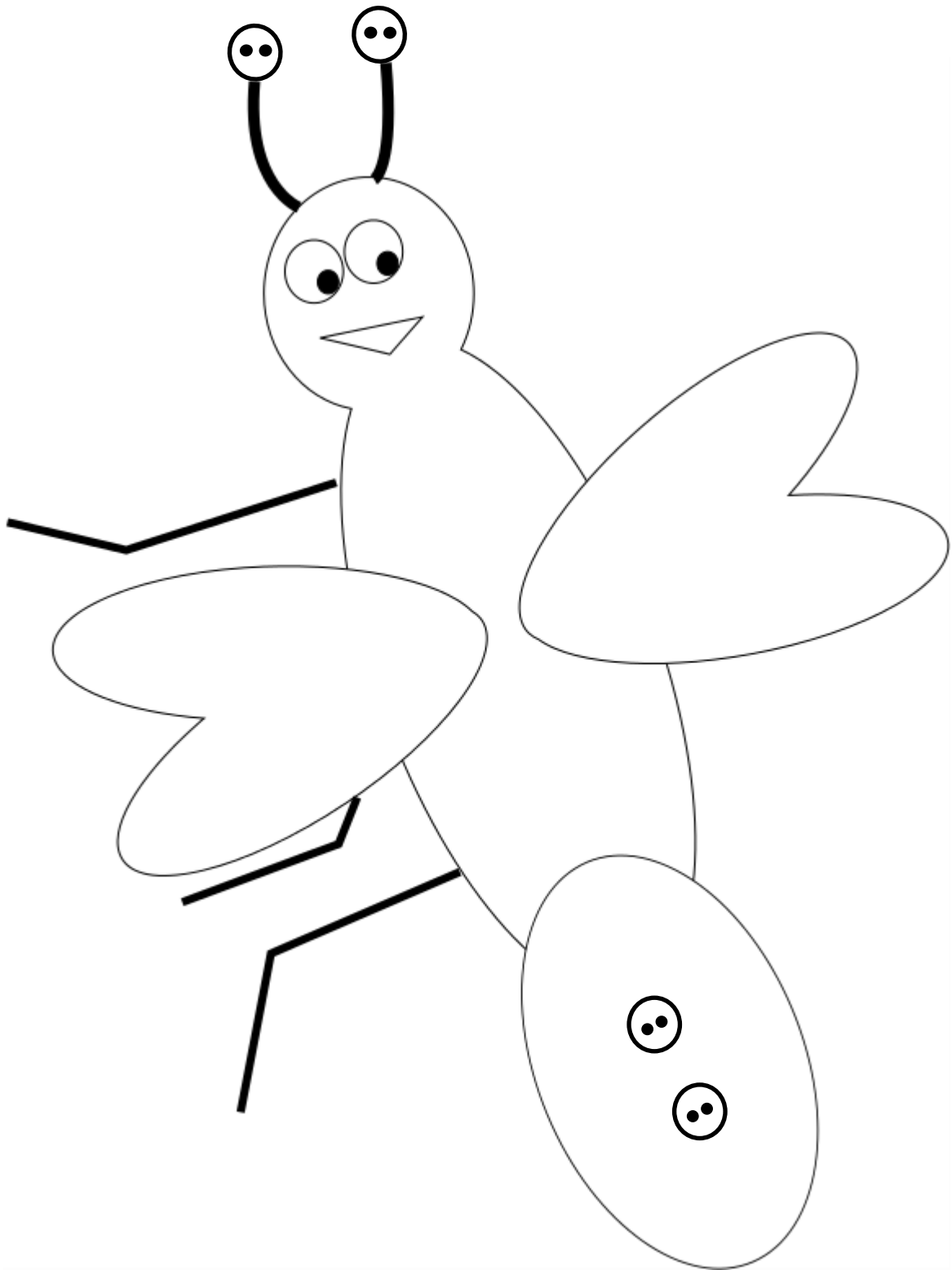


Solder Here



Tape Bridge





Try and create your own circuit so each LED is controlled individually!

