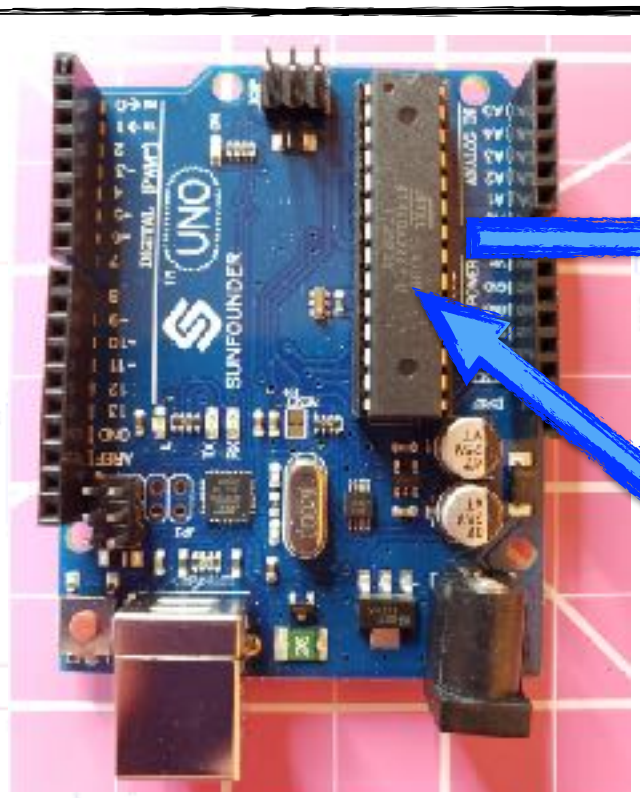
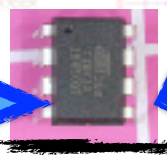
Big Chip and Baby Chips

In this chapter we're going to leave the basics behind. Take a moment to pick up the naked Arduino board you've been using over the last seven chapters in this book. Take a close look at the microchip the dominates the center of the board between the input pins the output pins. Can you see the ship is not actually soldered to the board? The chip is actually pressed into a socket which is soldered into the board. Your Arduino circuit board is actually just a really clever and well thought out package that houses a very common ship called the "ATmega328P-PU". This chip is actually found in a myriad of every day consumer and industrial machines. Appliances, 3-D printers in traffic automation systems all use variants of this chip. Why do you want to know about the variants of this chip? The official Arduino Uno R3 cost about $20. Generic copies of the board can be found for about $5. The ATmega328P-PU chip by itself retails for about $3. What if you don't need all of the features on this chip and you don't need all the components on the Arduino board? There are so many variants of this chip, some more sophisticated and some simpler, why not choose the chip that's right for you? In this chapter we're going to take a huge leap forward, further than other books have been written on the Arduino. We're going to take a bold and brave step into the world of the professional engineer and source our own microchip for $1, then use it to control a servomotor. If we can jump this hurdle how much freer are you to try riskier and more creative projects when all this on the line is a $1 microchip and a $1 servomotor?

We are also going to take and actually build our own circuitboard! Now that you're looking at the microchip as the actual functional heart of the Arduino Uno, then what is rest of the board really do? If are going over simplify just a bit, I would argue that most of rest of the board is a set of convenient plugs that pins call pins that make it easy the plug wires in. The USB plug is also serving that same function. When we program within the Arduino IDE we're simply sending data through a cable and that eventually connects to various pins on the microchip. There's some hardware between into the doesn't translation, just like a some hardware behind the barrel connector that allows us to use a wide range of voltages to feed the Arduino. Still, everything seems to end up connected to the microchip. Why don't we start instead with requirements for our project and then select the simplest possible chip mounted on the simplest possible circuit board, that will do the job that we have envisioned?

What you see below is a result of a lot of exploration and a lot of collaboration with folks at Concord.org and PaperMech.net as well as the hard work of Susan Klimczak. The chip we are all focused on is called the ATTINY85, a $1 microcontroller found in so many consumer and Industrial products from alarm clocks to assembly line automation. My contribution here has been around the paper egg circuit board shown below. I have included enough information here to get started with the ATTINY85, but I would also encourage you to google "Susan Klimczak ATTiny Adventures" for a page with more detail that is outside of the scope of this chapter as your interests and project aspirations expand.

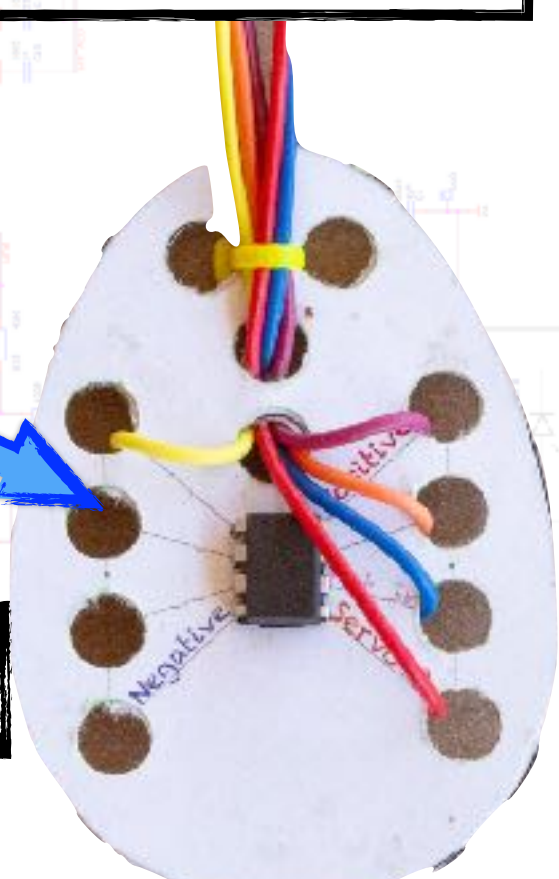




ATTiny85

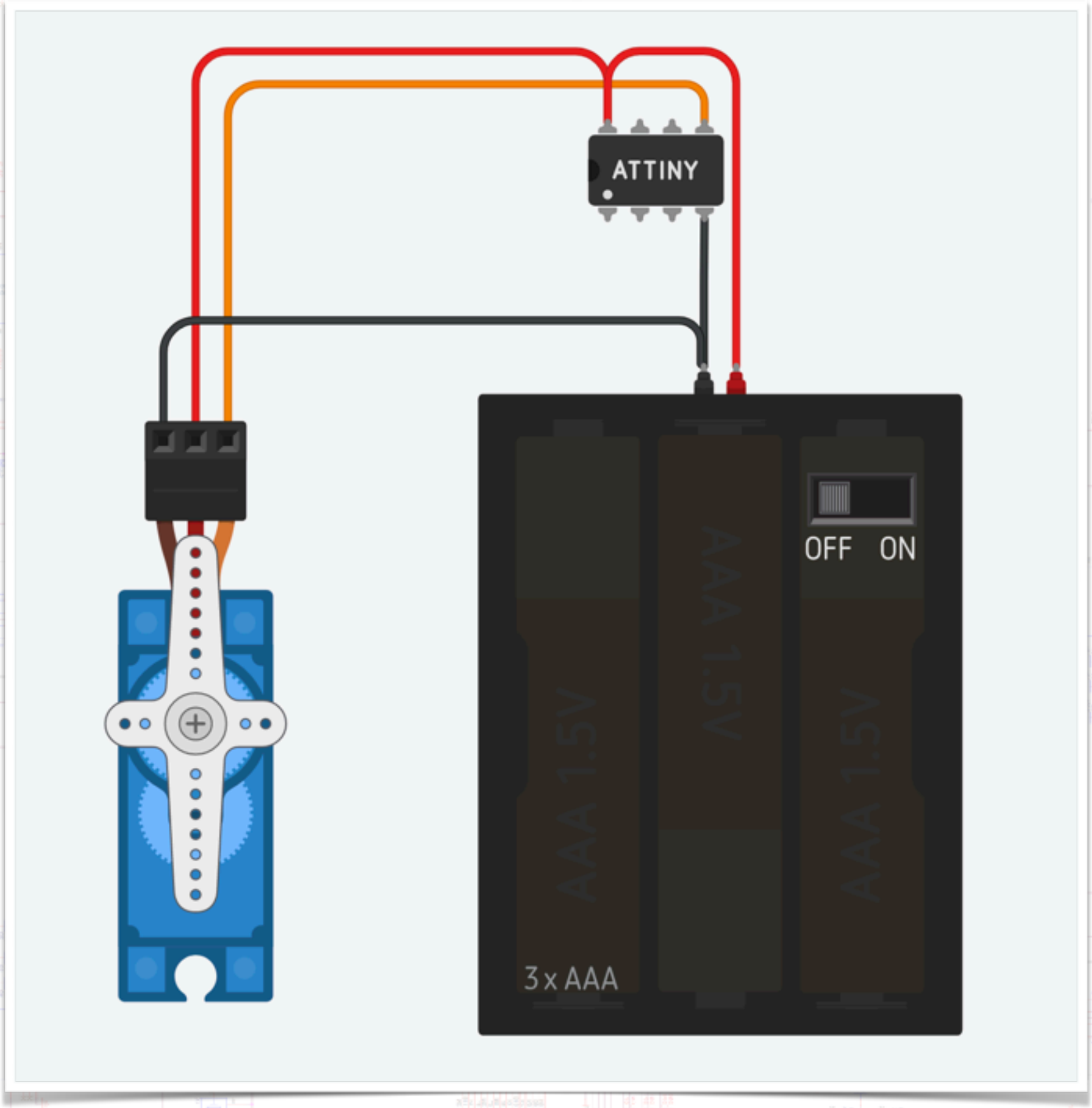
ATmega328P-PU

ATTiny85 mounted on a paper egg



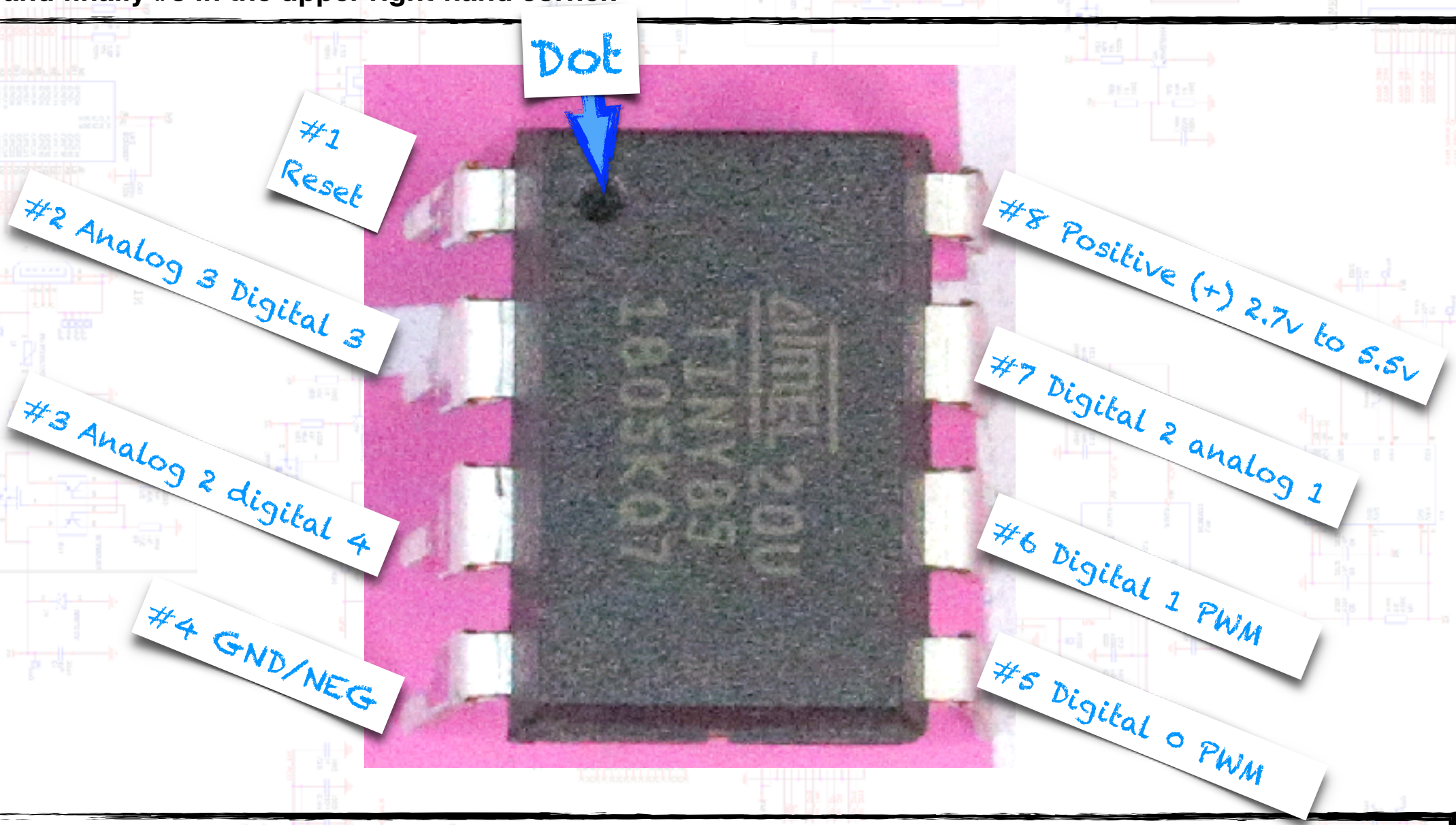
ATTINY85 on Tinkercad Circuits

Tinkercad circuits is a great place to go to get started with this chip. I hope you started to explore more of the starters and examples that are available on the Tinkercad circuits website. This is one of those rare free and successful community websites where people from all backgrounds and interests post circuit designs and code that have worked for them for free and without ads. “Soft Servo Sweep” is one of many excellent examples on the site. Below is the circuit diagram for the wiring setup. You can use 3AA batteries or simply run jumpers from the Arduino Uno’s 5v and GND pins to use as a power source. Once the chip is programed all this is a power, ground and a single single wire on physical pin #5!



Translating the PINS

My hope is that in previous chapters you would have become quite comfortable at hooking up various pins and relating that physical choice to our designation within software to control those physical pins. The only difference now is her throwing away all of the stuff between where we were plugging in a jumper wire on the Arduino to now connecting directly to an actual pin on a chip. You will see shortly that we will actually use a socket to extend the pen so we can wrap the wire around it. Before we get to that point though, we need to understand the layout of the physical pins on the chip. With an engineering circles this is called getting a “pinout” which is just another word for defining which pin has which number assigned to it. All chips that I know of look symmetrical and have ridiculously small writing on them which is almost illegible unless under extreme magnification. Having said that every chip in this family (and almost every other chip that I know of) has an orientation Dot. This is literally a small indent on the chipper self which is much easier to see in the faint labeling. A simple rule is when you find dot on the chip oriented chip so the dot is in the upper left-hand corner. From this orientation the physical numbering of the pins starts at #1 in the upper left corner and proceeds counter clockwise from there. In this case when the chip has eight total pins on it we are starting with #1, then below that #2, #3 down to #4 in the lower left corner, then around to #5 in the lower right corner and up to the #6, #7 and finally #8 in the upper right-hand corner.



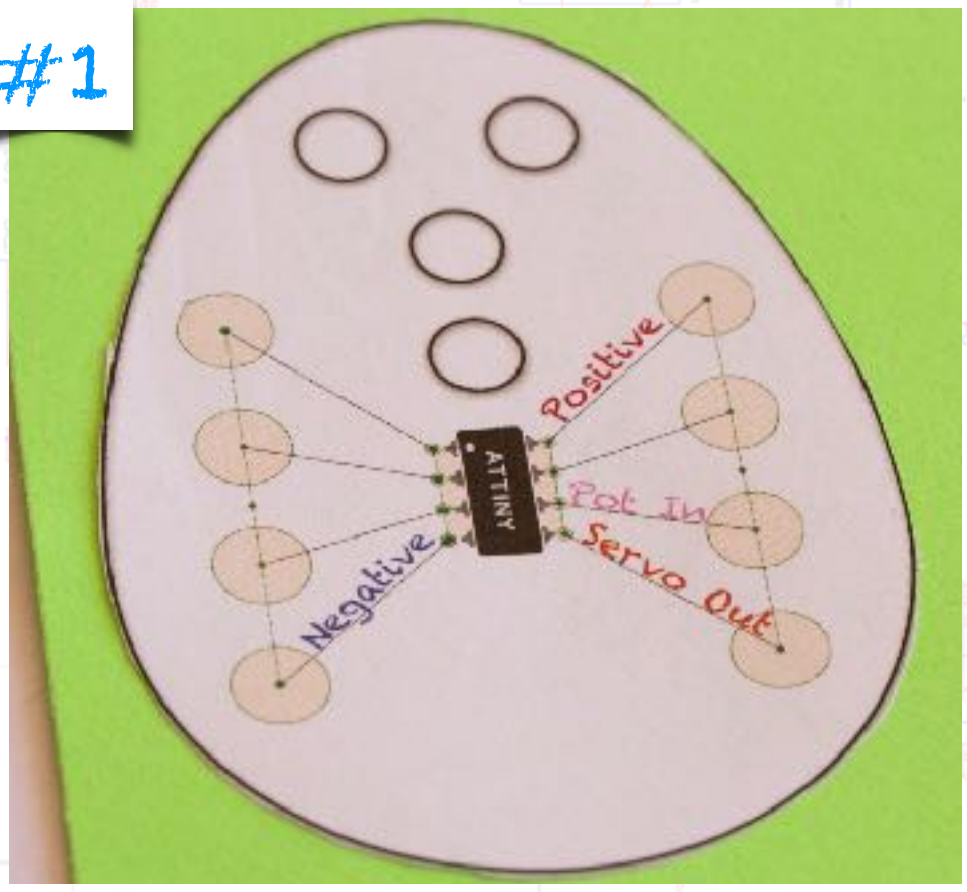
My hope is that your instinct at this point after reading the first part of the book is to first find the power (+) and ground (-) pins and then assume everything else is the signal pin. In this case look at #8 for power and #4 for ground. Since we are going to use serve a motor in this project want to find pins that have a PWM function (PWM stands for Pulse Width Modulation, basically it is a way to encode a range of values that the servo can use to translate to a certain number of degrees). In the case of the ATTINY85 we have pins #5 and #6. The rest of the label on these two pins refers to what number our program needs to use to send a signal to the physical pin. Let’s use pysical pin #0 and to do that we would attach the software naming the servomotor to pin #0. There are some other very interesting signal pins on this chip. The physical pins #2 and #3 can be used for analog inputs from things like a moisture sensor or a light sensor. The physical #1 pin will reset the program every time is gets a signal. With all the Arduino experience that I have I can say that 95% of the projects of ever undertaken could really have been done with the single $1 microcontroller.

Making a Paper Circuit Board

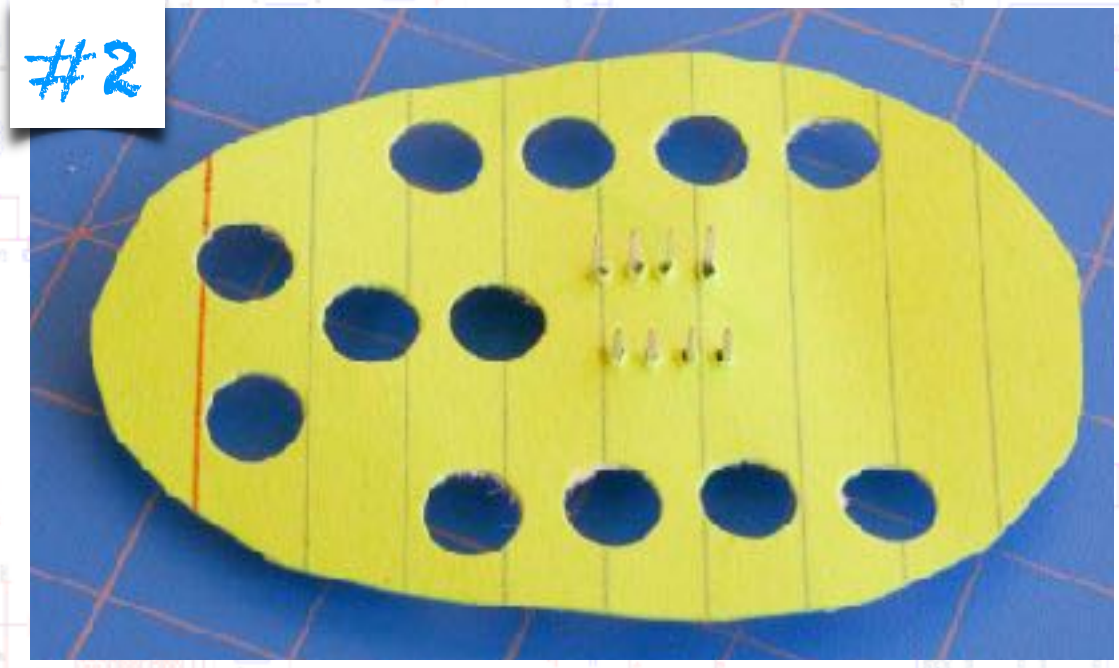
What we are trying to do with this paper circuit board is provide a substrate that is suitable to thread wires in and out of it. All all of the connections to and from the circuit board will be made with these jumper wires sticking up from the circuit board surface. The benefit here is it due to the fact the wires are flexible in reference to chip. This is a much more durable design when dropped.

Step #1) Use a gluestick to secure a paper design to an index card or cardstock. Place the ATTINY85 chip on a piece of paper. Lightly press the chip into the paper and leaving indentations in the paper where you can follow with a pushpin and punch holes for the legs. Punch several 1/4" holes around the perimeter and towards the center of the paper. I designed an egg shape. You might have a much more creative animal or letter shape. I printed the graphics for my design in OSX Keynote. Overall the goal is to provide a semi-rigid surface to become sandwiched between the ATTINY85 and the socket. Once you punch holes for the legs with a pushpin then #2) insert the ATTINY85 chip into the paper. #3) Place the socket under the chip and push down gently. I did this by actually holding the socket in my left hand and holding the chip in my right hand. #4) show the socket ready to wire. With this test fit done remove the chip from the socket again and proceed with wire wrapping on the next page.

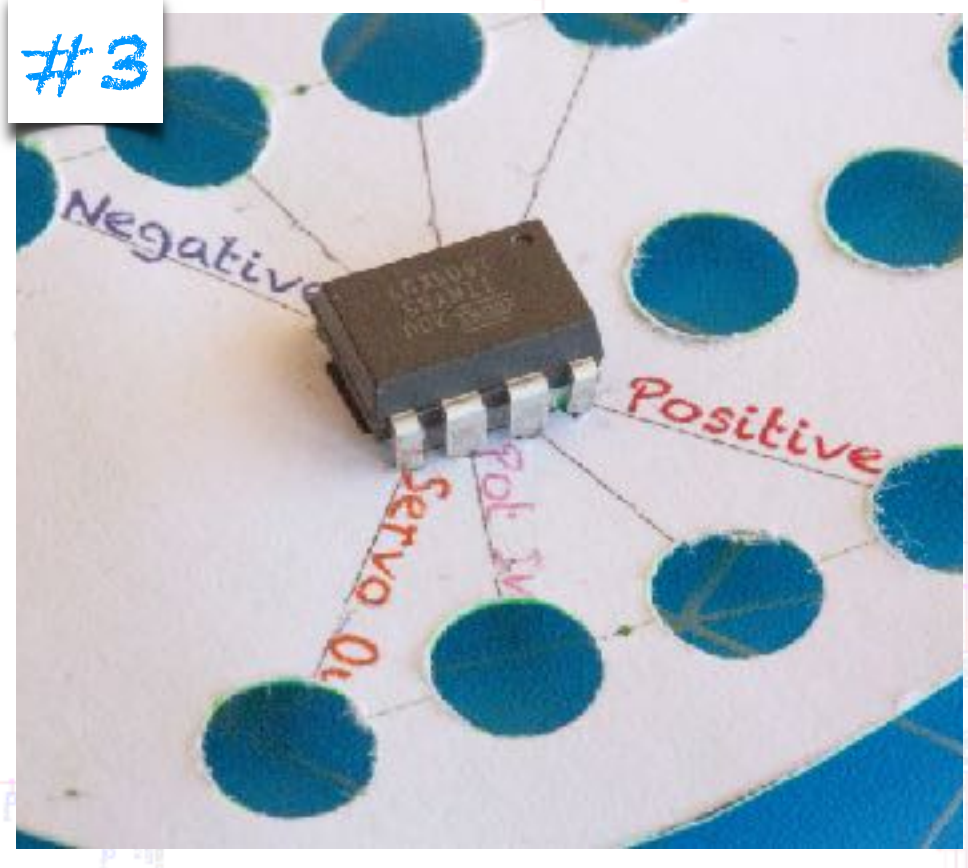
#1



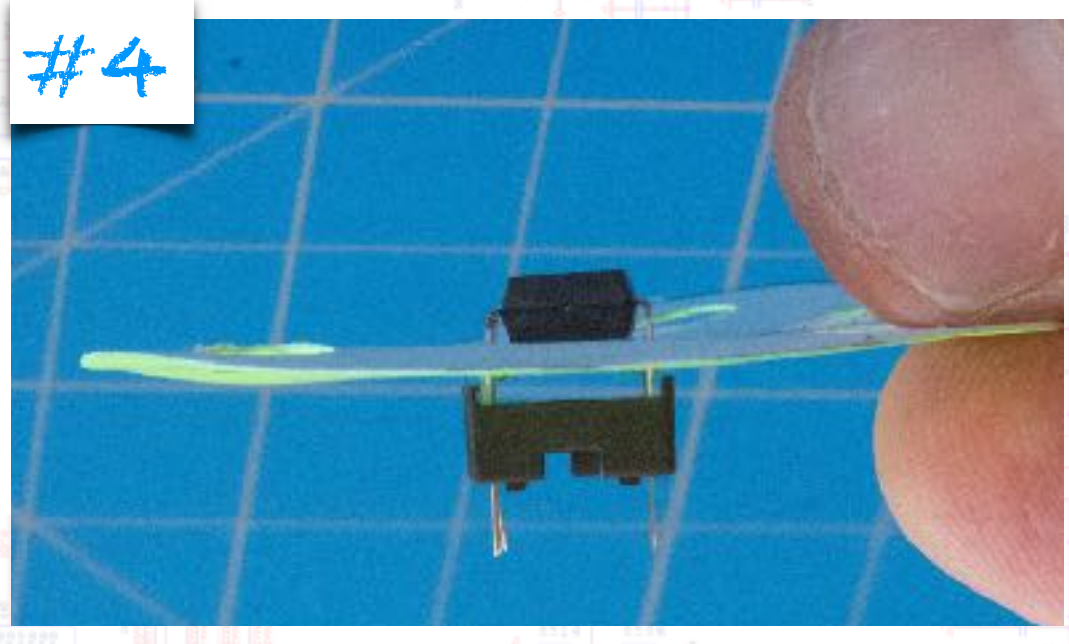
#2



#3



#4

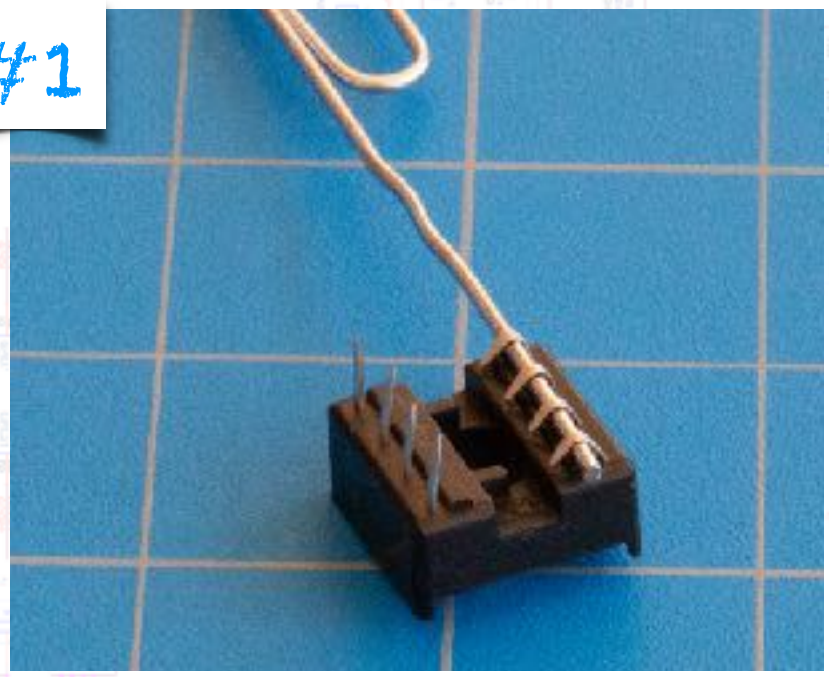


Advanced Wire Wrapping

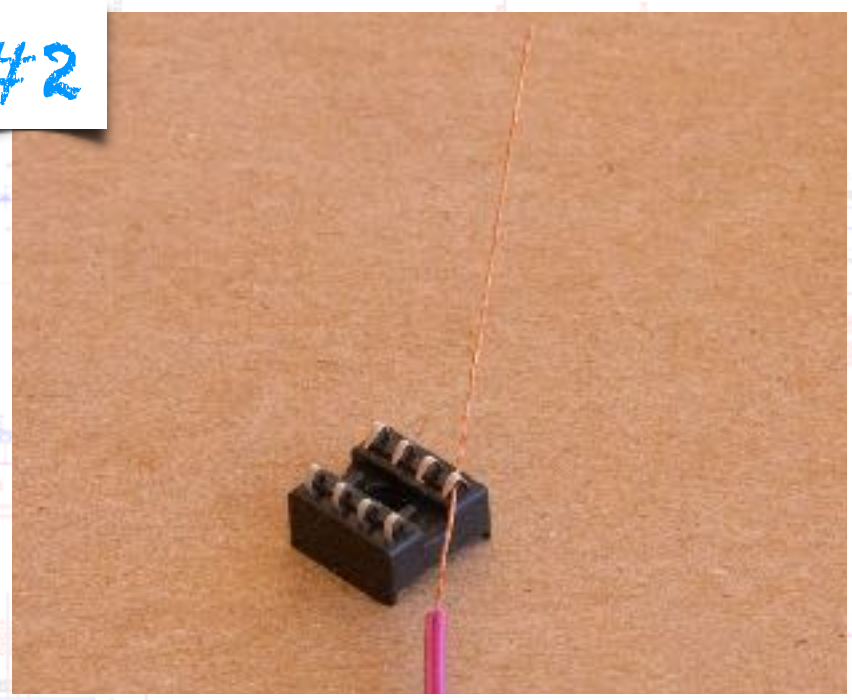
In past chapters of learned to on the Neopixel board and to the ends of jumper wires. I want to teach you wire wrapping instead of soldering because I want these project you want to build to come together as quickly as possible so you're more confident iterating on them quickly. Soldering is a great skill, but it can slow you down here. Follow the steps below to wire wrap the socket that comes with the ATTINY85 chip (make sure you order the chip with flat leg sockets included as round leg sockets will not work for wire wrapping.

Step #1) Use a large paperclip as cylindrical base to bend the socket legs over toward the center. #2) After you have done step #1 to both rows of four legs then insert a wire with 1.5” of insulation stripped of and the tip twisted. Insert that wire in the loop from the outside-center-bottem of the bent over socket leg. #3) wrap the wire back around its self a few timer until secure. #4 trim off the excess length with scissors. #5) repeat for socket legs corresponding to physical ship legs #8-#5 and #4

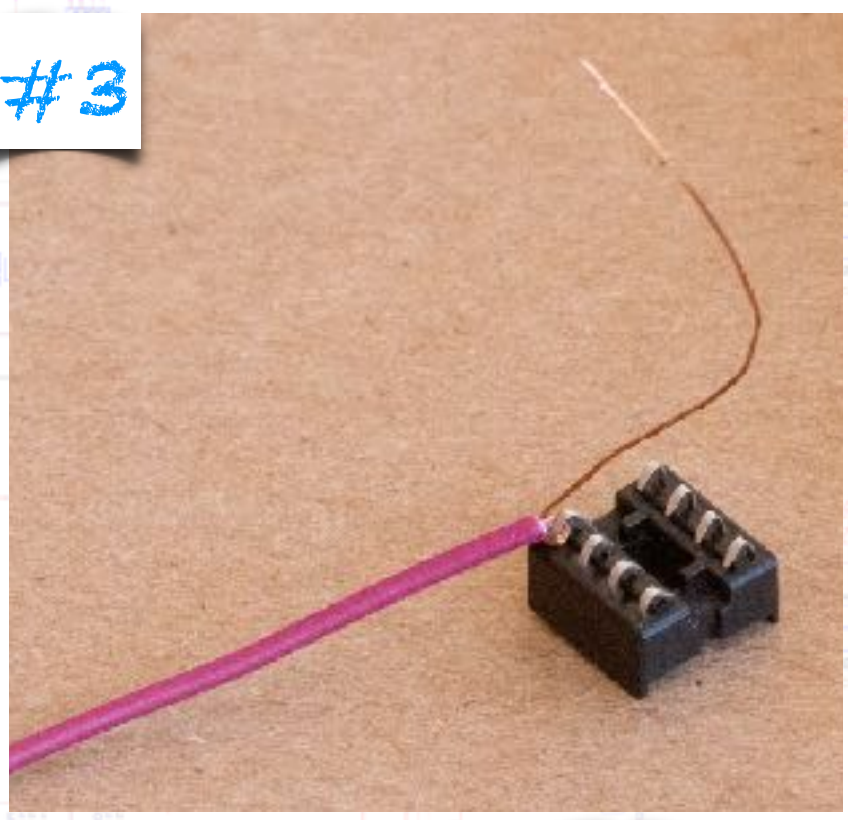
#1



#2

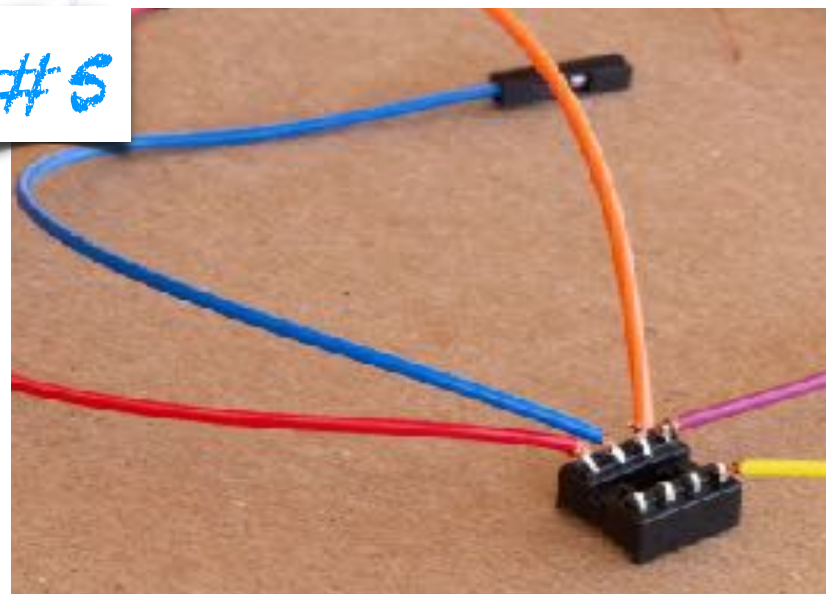


#3

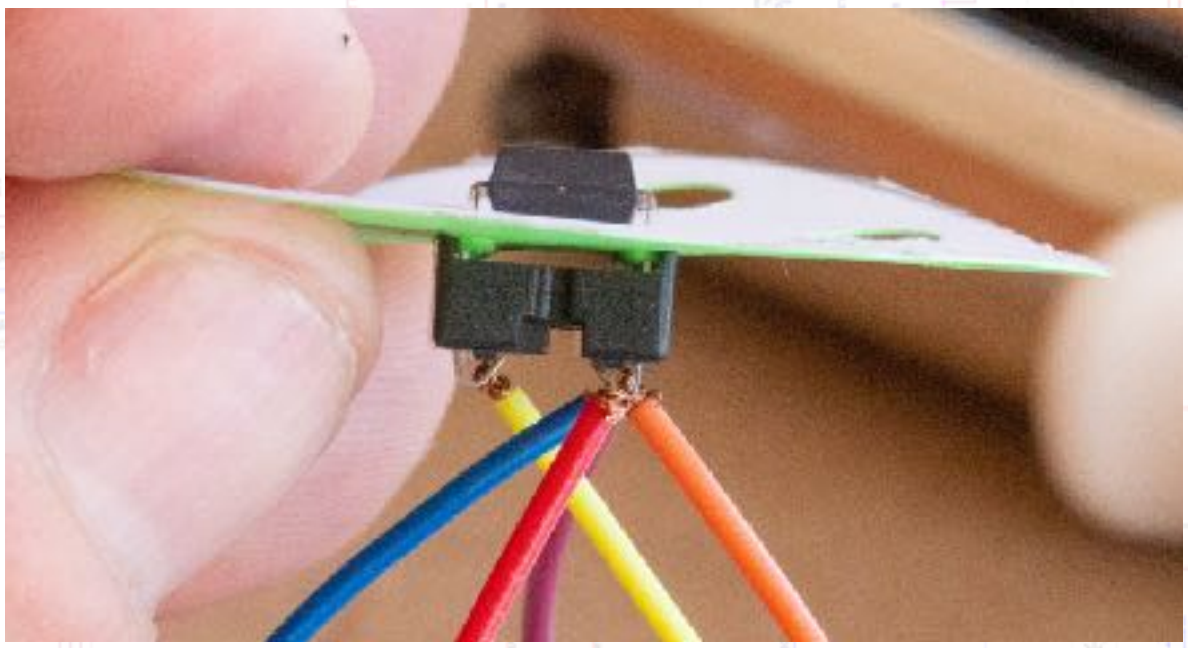


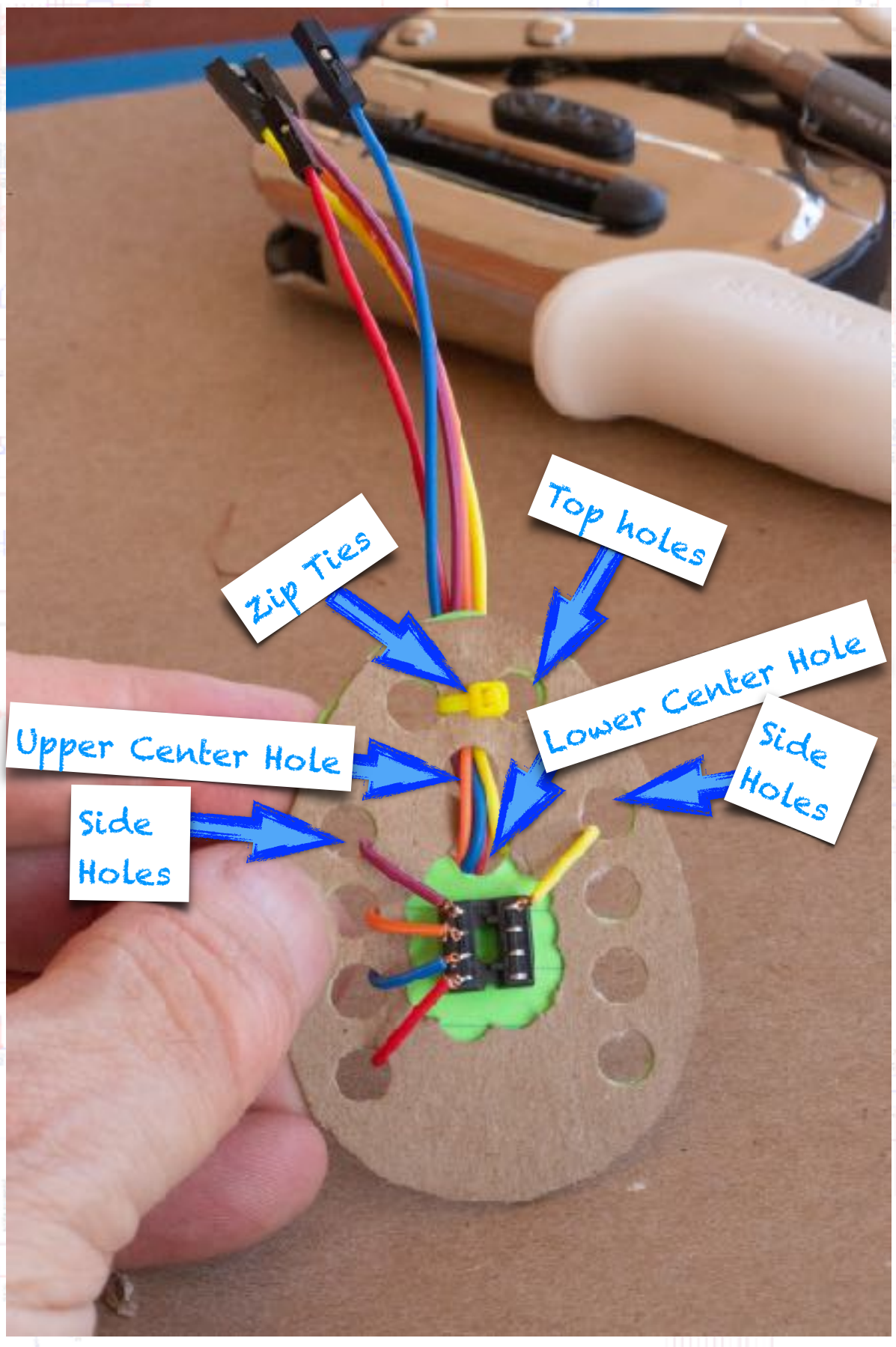
#4

#5



Adding Tension to the Wire

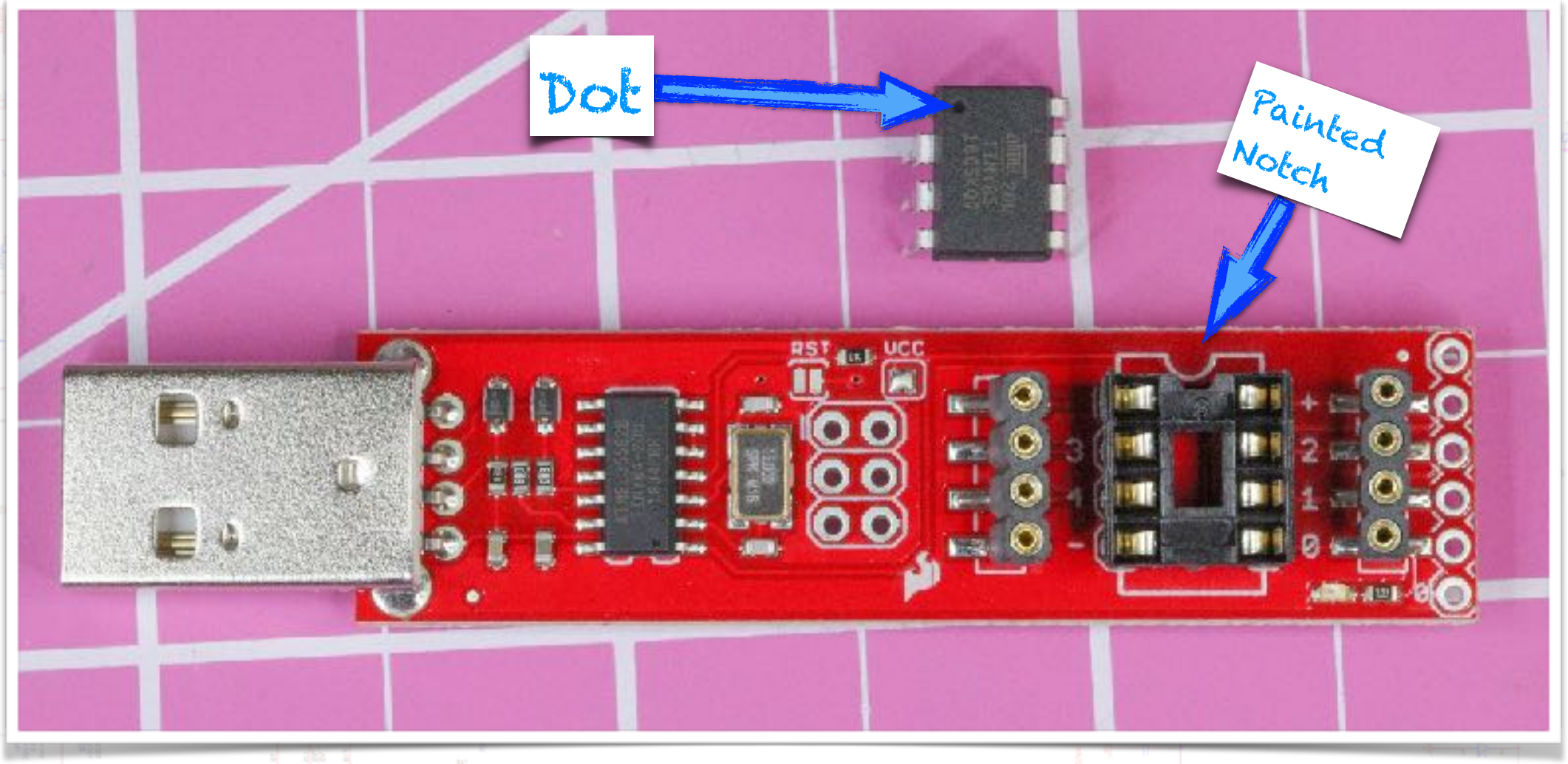




Start with the visual of why this wire wrapping technique works: place both of your hands in front of you, put the tip of you thumb on the tip of the index finger on the same hand. You should be looking down at two circles. Open in the gap between your right thumb and index finger, then move to your left hand closing the gap between your thumb and your index finger inside of the other circle so you now have to interlaced circles. This is much like the electrical connection we just made when we wrapped wire around the bent over socket legs. There can be a small amount of room between the wire and the socket leg which runs the risk of an incomplete electrical connection. The purpose of the holes in the paper egg and wire tie on top is to permanently add attentional pulling tension in the wire towards the outside of the socket just like if you try to separate your hands now while keep them under tension there will always be contact between your left and right hand because of that outward tension. The picture on the top left shows a wires dangling from the sockets ready to be threaded through the punched holes. The photo on the lower left shows the completed route with the wires 1st threaded up through the holes on the side, then down through the lower center hole, then up through the middle center hole and finally zip-tied between the two top holes. This is all to keep tension on the wires - always gently pulling them away from the socket legs an insuring a secure connection.

Programming the ATTINY85

How do you program the ATTINY85 if there is no USB port? The answer is you need a USB port. There is a more complicated way to attach the ATTINY85 to a breadboard, then hook it up to an Arduino and then passed data between the computer through the Arduino an into the ATTINY85. For now we will use the "Tiny AVR Programmer" from Sparkfun. This is a $20 dollar USB stick where we can mount our ATTINY85 temporarily programming. This is quite an easy process once you get through the initial setup which does take a little focus. For right now take a ATTINY85 chip and locate dot and orient it so the dot is in the upper left corner. Then take the Tiny AVR Programmer and orient it so the USB is on the left. Locate the white line around the chip socket and look for the notch in the top. Very gently squeeze all eight legs of the chip between your thumb in your forefinger narrowing the gap between the two rows of legs by about 1 mm. While the chip is between your fingers align the eight legs with the socket and pushed directly down. My only caution here is to always do this when the USB stick isn't one hand in the chip is in the other hand And nothing is attached to the computer. This is not because inserted the chip while the USB board is attached to the computer will heard anything, but it is because the force needed to place the chip correctly in the socket might vendor break to USB stick. After this is done insert the USB stick into the computer before you start programming (caution: inserting the USB stick in after you start changing settings within the Arduino IDE will fail to recognize the USB stick).



Getting the Arduino IDE to work with the ATINY85

Below are the steps to get the ATTINY85 working with the Arduino IDE. Additional instructions are needed for Windows and appear below the below steps.

#1 **Plug in the ATTINY85 to the Tiny AVR Programer, do this first!**

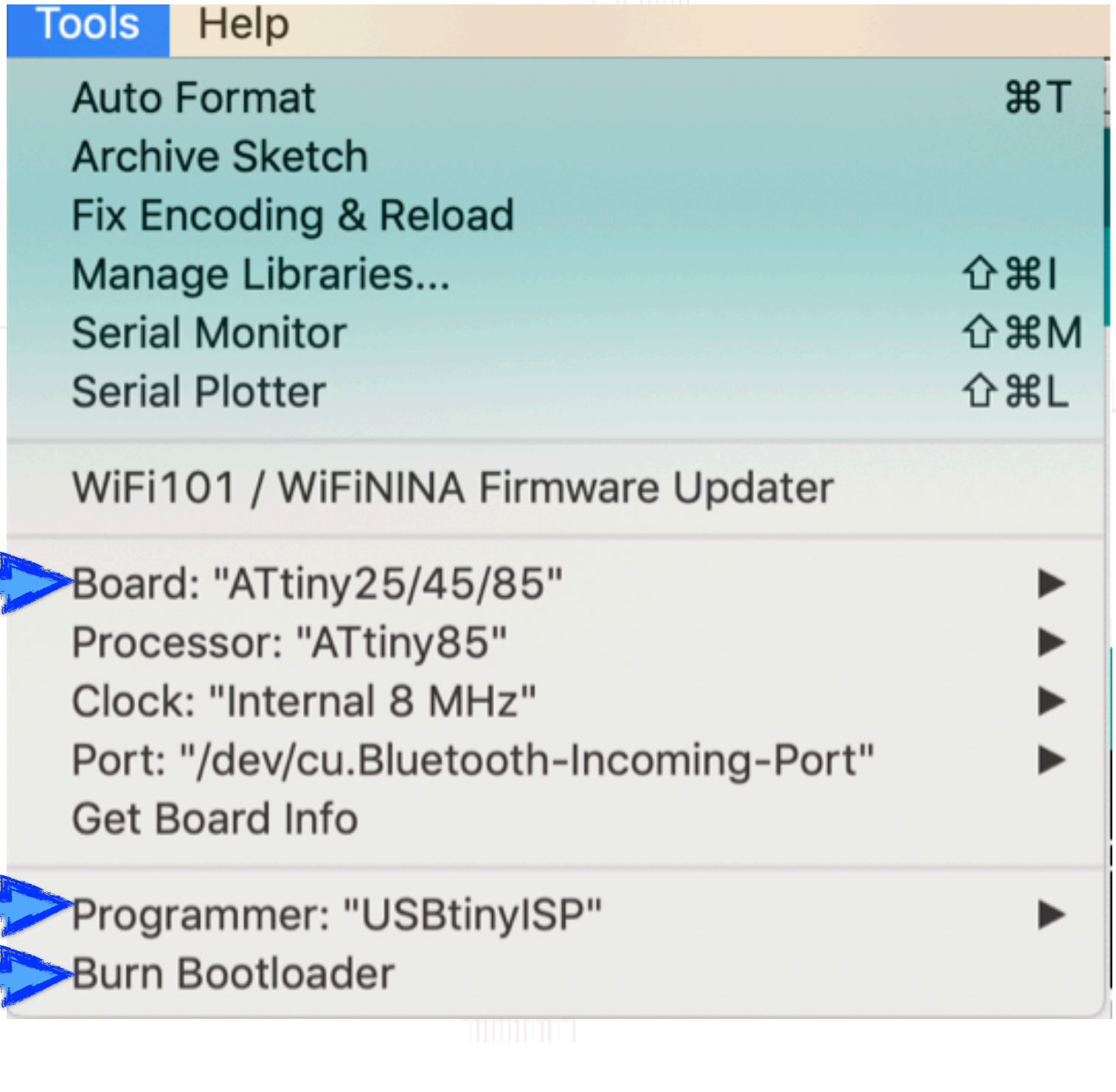
#2 **Add the chip in the Tools > Boards > Board Manager > Search ATINY > Be sure you plug your ATTiny**

#3 **Tools > Board > ATtiny85 (8MHz internal clock)”**

#4 **Tools > Programmer > USBTinyISP**

#5 **Tools > Burn Bootloader (this needs to be done every time you plug in a new ATTINY85 (a chip you have never programed before). The above steps only need to be done once.**

Windows: the manufacturer of the Tiny AVR Programer has the best instructions for the Windows operating system here: https://learn.sparkfun.com/tutorials/tiny-avr-programmer-hookup-guide/all



Uploading Code to the ATTINy85

Below is a simple test program that will blink the LED of the ATTINY85 every half second while it is still on the Tiny AVR usb stick. It will start blinking as soon as the program is uploaded. Proceed to Tinkercad Circuits for other designs like the Smooth Servo Sweep. Search by ATTINY85

ATTINY_test_on_TINY_AVR_USB_Stick | Arduino 1.8.9

ATTINY_test_on_TINY_AVR_USB_Stick

```
//This sketch is useful for testing whether or not your ATTiny85
//is uploading code.  It blinks the onboard LED on the TINY AVR
//USB Stick every half second after you upload the code.


int blinkPin = 0;    /* onboard LED is on pin 1

void setup()
{
  pinMode(blinkPin, OUTPUT);    /* onboard LED is an output
}

void loop()
{
  digitalWrite(blinkPin, HIGH);  /* Turn on LED  */
  delay(500);                    /* Wait half a second  */
  digitalWrite(blinkPin, LOW);   /* Turn off LED */
  delay(500);                    /* Wait half a second  */
}
```

Done Saving.

```
The sketch name had to be modified.
Sketch names must start with a letter or number, followed by lette
numbers, dashes, dots and underscores. Maximum length is 63 charac
```

3 ATTiny25/45/85, ATTiny25, Internal 8 MHz on /dev/cu.usbmodem14601