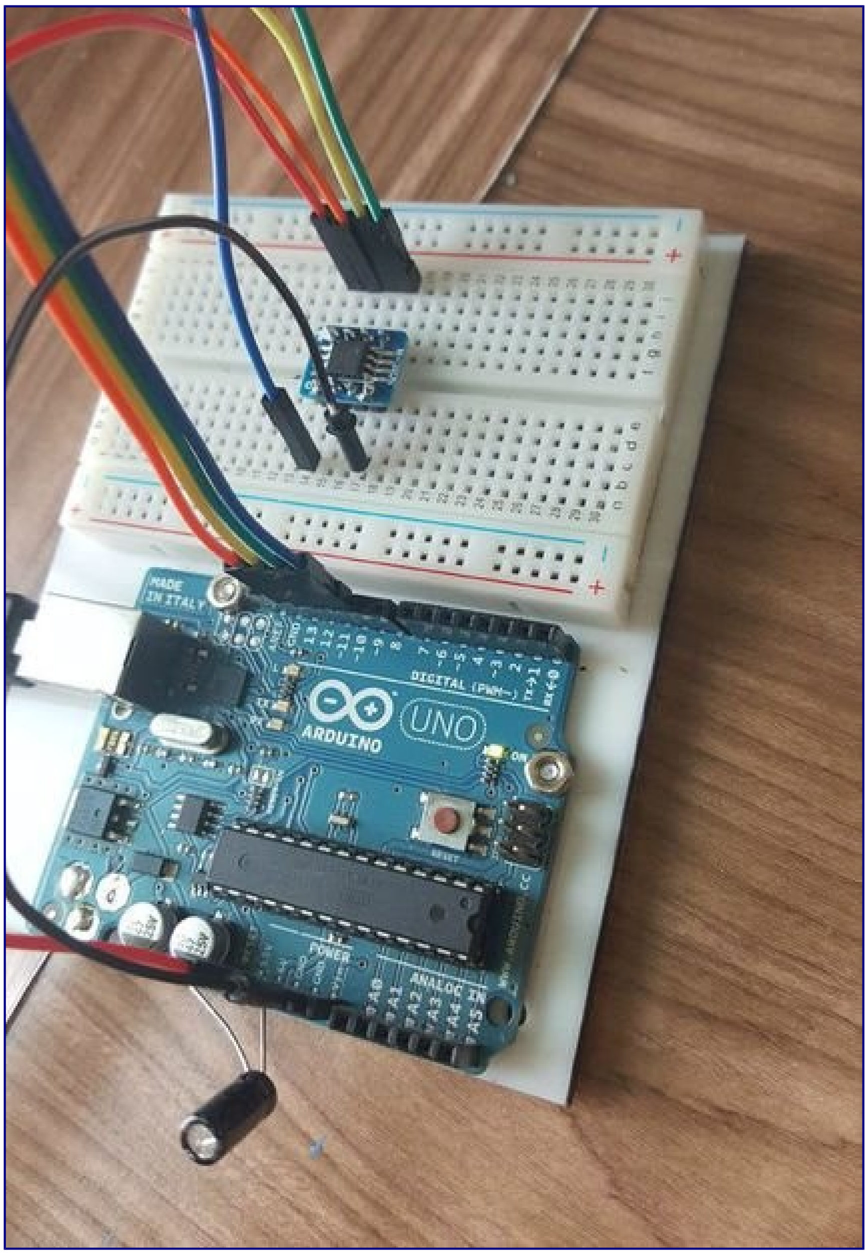


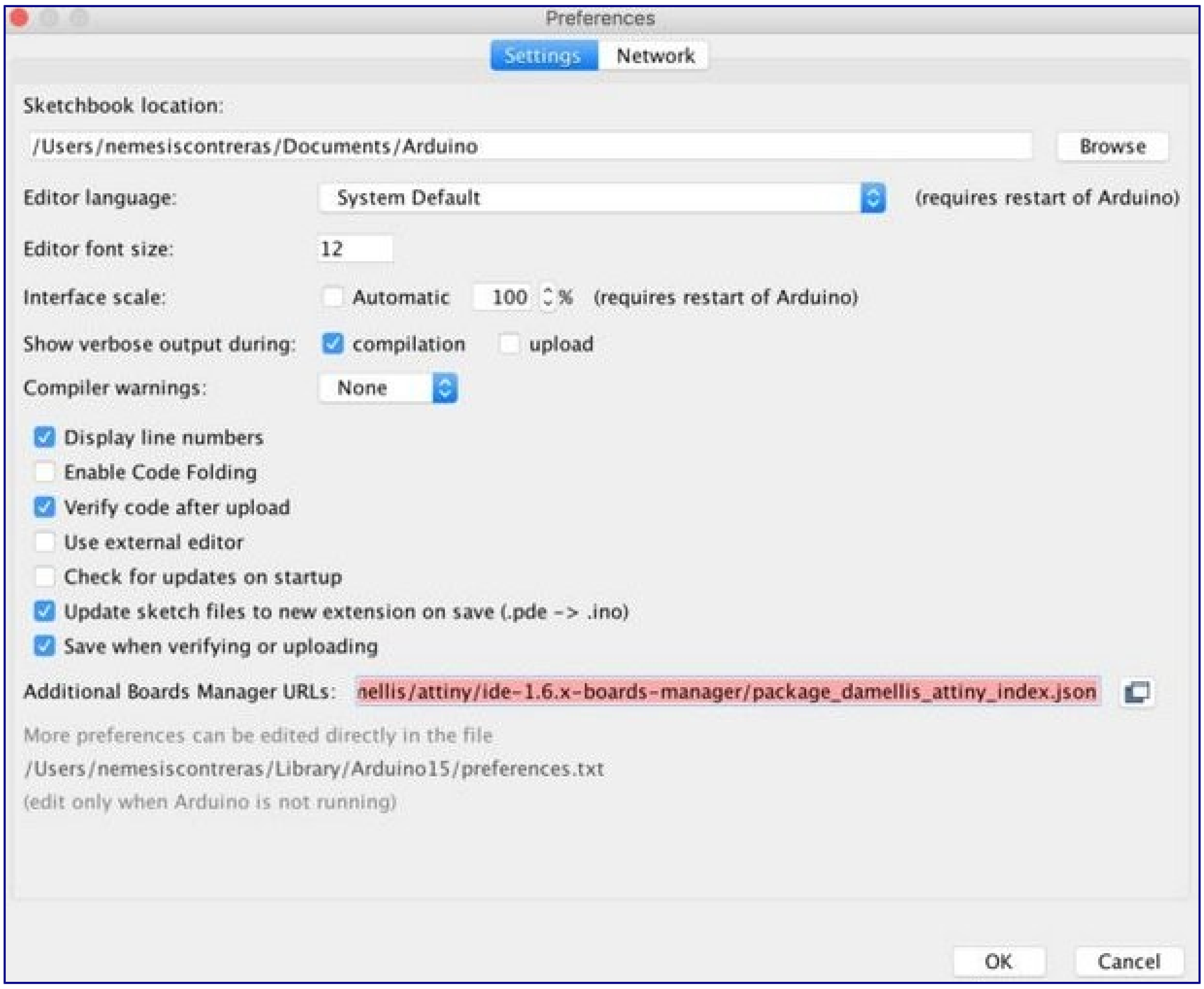
Quick tutorial showing how to program the ATtiny85 from the Arduino IDE with the help of the Arduino Uno!

This tutorial was requested by my friend Orlando so hope it helps !

Comments,Concerns,Feedback,Requests welcomed:

@NemesisContrer8

Step 1: Add support for the ATtiny85 to the Arduino URL Board Manager


Preferences

Settings | Network

Sketchbook location: /Users/nemesiscontreras/Documents/Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

Interface scale: Automatic 100 % (requires restart of Arduino)

Show verbose output during: ✓ compilation ☐ upload

Compiler warnings: None

✓ Display line numbers

☐ Enable Code Folding

✓ Verify code after upload

☐ Use external editor

☐ Check for updates on startup

✓ Update sketch files to new extension on save (.pde -> .ino)

✓ Save when verifying or uploading

Additional Boards Manager URLs: nellis/attiny/ide-1.6.x-boards-manager/package_damellis_attiny_index.json

More preferences can be edited directly in the file

/Users/nemesiscontreras/Library/Arduino15/preferences.txt

(edit only when Arduino is not running)

By default the Arduino IDE does not support the ATtiny85 it's required to add support for the Attiny85 to the Arduino Board Manager:

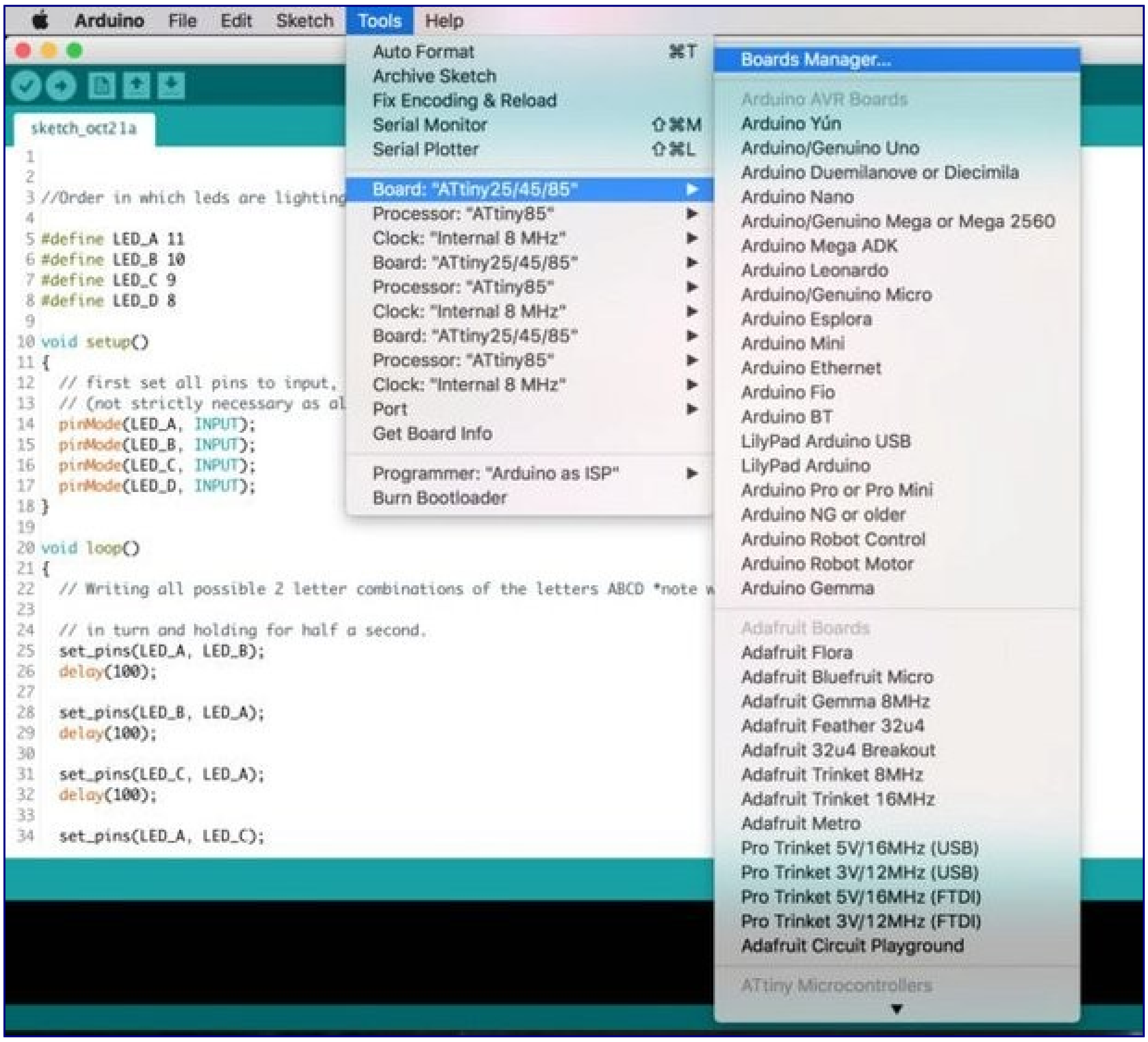
- From the Arduino IDE Go to Arduino->Preferences then scroll down to *Additional Board Managers URLs*
- Copy & paste the following (if you already have a board manager URL just add a comma before pasting)

Thanks David-one of the Arduino founders for writing the code!

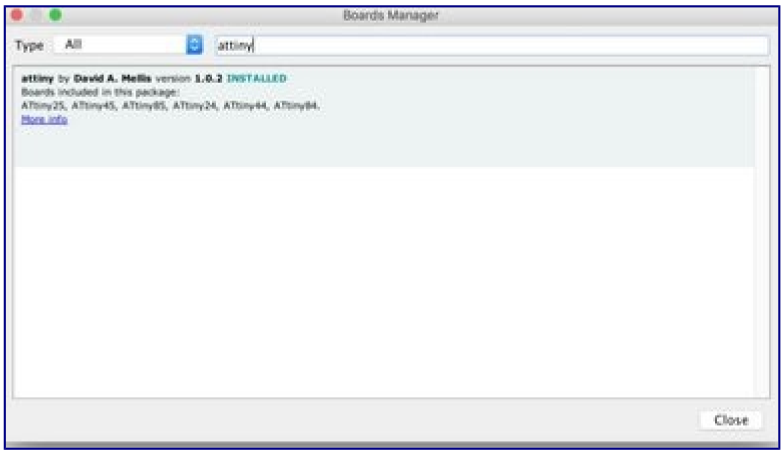
```
<p>https://raw.githubusercontent.com/damellis/attiny/ide-1.6.x-boards-
manager/package_damellis_attiny_index.json</p>
```

- Press "OK" at the bottom then restart the Arduino IDE

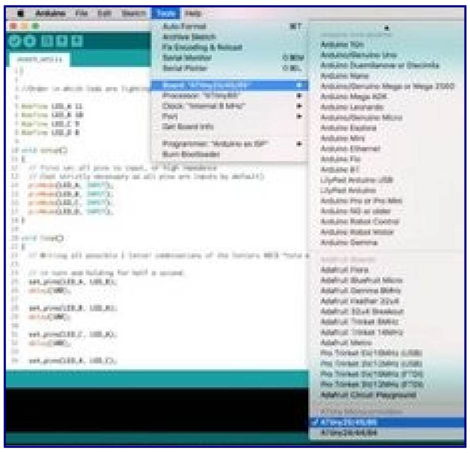
Step 2: Install the ATtiny board package



```


//Order in which leds are lighting

#define LED_A 11
#define LED_B 10
#define LED_C 9
#define LED_D 8

void setup()
{
  // first set all pins to input,
  // (not strictly necessary as al
  pinMode(LED_A, INPUT);
  pinMode(LED_B, INPUT);
  pinMode(LED_C, INPUT);
  pinMode(LED_D, INPUT);
}

void loop()
{
  // Writing all possible 2 letter combinations of the letters ABCD *note w

  // in turn and holding for half a second.
  set_pins(LED_A, LED_B);
  delay(100);

  set_pins(LED_B, LED_A);
  delay(100);

  set_pins(LED_C, LED_A);
  delay(100);

  set_pins(LED_A, LED_C);
```

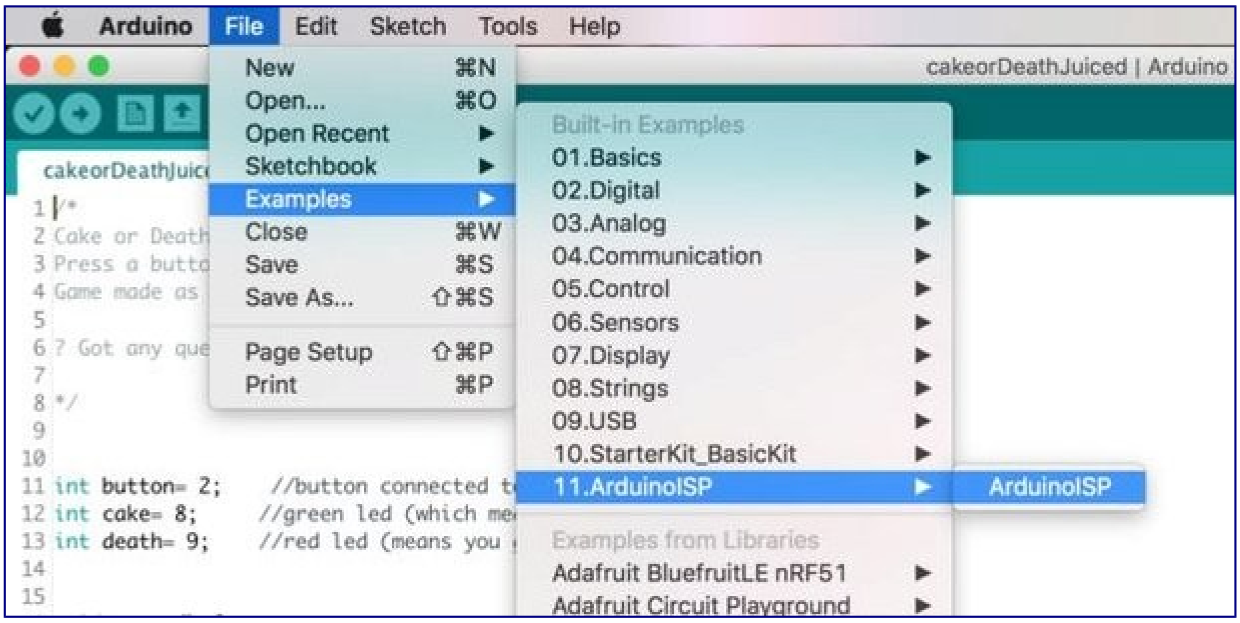


attiny by **David A. Mellis** version **1.0.2** INSTALLED
Boards included in this package:
ATtiny25, ATtiny45, ATtiny85, ATtiny24, ATtiny44, ATtiny84.
More info



- From the Arduino IDE go to Tools--> Board-->Boards Manager
- A new tab will open and at the top of the tab type: *attiny*
- Select Install on the Attiny by David. A Mellis
- Restart the Arduino IDE
- The ATtiny85 board should now be added ! Go to Tools--> Board-->Attiny85

Step 3: Set the Arduino Uno into ISP mode



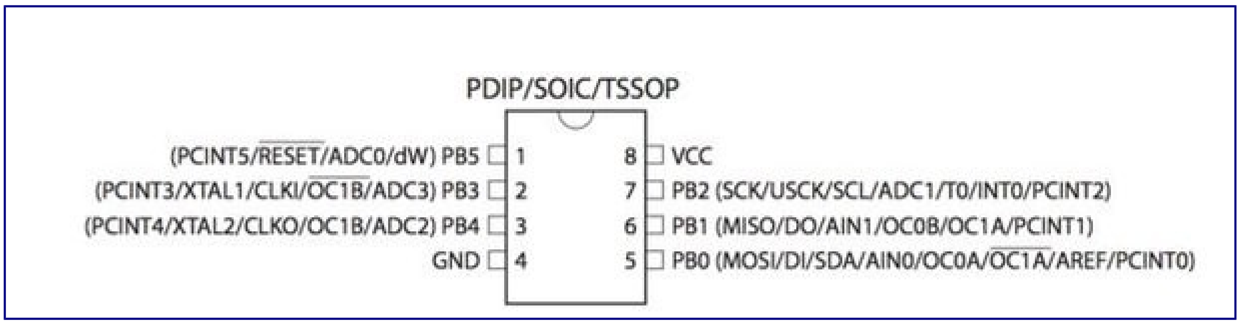
```
/*
Cake or Death
Press a butto
Game made as

? Got any que

*/


int button= 2;     //button connected t
int cake= 8;      //green led (which me
int death= 9;     //red led (means you
```

Since what we want is to be able to program the ATtiny85 from the Arduino IDE which requires to burn the bootloader to the ATtiny85 we will need to "prep" the Arduino fist by uploading the ISP sketch to it.

In the Arduino IDE select File-->Examples--> 11. Arduino ISP-->ArduinoISP

the ISP sketch should open and upload it to your Arduino Uno

Step 4: How a Microcontrollers Pins Are Labeled


PDIP/SOIC/TSSOP
(PCINT5/RESET/ADC0/dW) PB5 1
(PCINT3/XTAL1/CLKI/OC1B/ADC3) PB3 2
(PCINT4/XTAL2/CLKO/OC1B/ADC2) PB4 3
GND 4
8 VCC
7 PB2 (SCK/USCK/SCL/ADC1/T0/INT0/PCINT2)
6 PB1 (MISO/DO/AIN1/OC0B/OC1A/PCINT1)
5 PB0 (MOSI/DI/SDA/AIN0/OC0A/OC1A/AREF/PCINT0)

Before the connections are made there is a very important fact to know how pins on microcrontrollers/ICs are labeled.

Pin numbers used to program a chip on the Arduino IDE are based on how the chip manufacturer has internally named/aranged the pins . The manufacturer of the ATtiny85 is ATMEL (the AT in **AT**tiny85-actually stands for **AT**MEL);It's common for chips to have the first two initials of the company who makes them.

Pins are gathered into groups called "ports" these ports are labeled A,B,C etc. Each port has a number of pins which are labeled 0,1,2,3 etc and stick out on different parts of the chip which is why a **microcontroller's physical pin often time will be different than the pin number used when programming the chip.**

An example:

PB0 (in the above datasheet) just means pin 0 is located on Port B of the Chip.

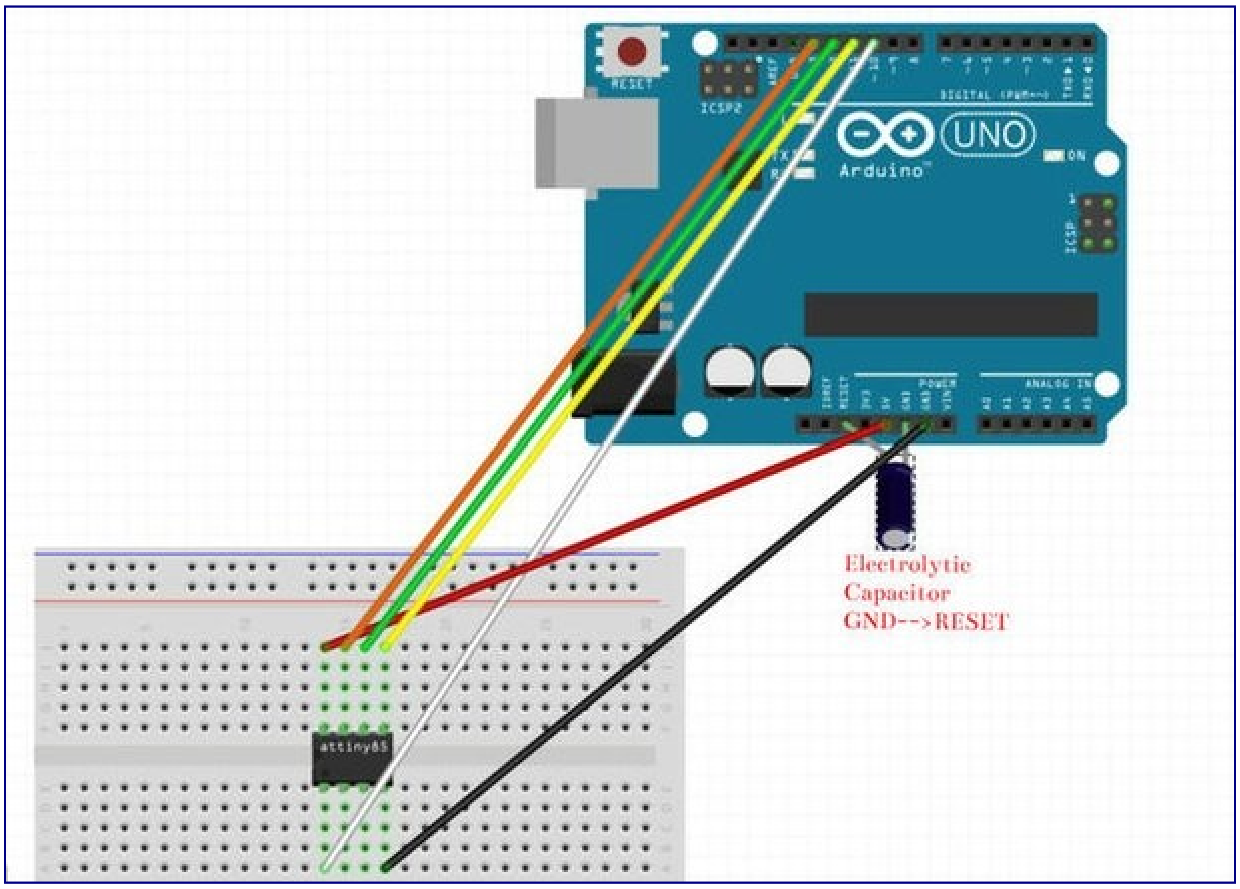
If pin 0 was located on Port A the name would look something like PA0 (**P**ort **A** pin **0**)

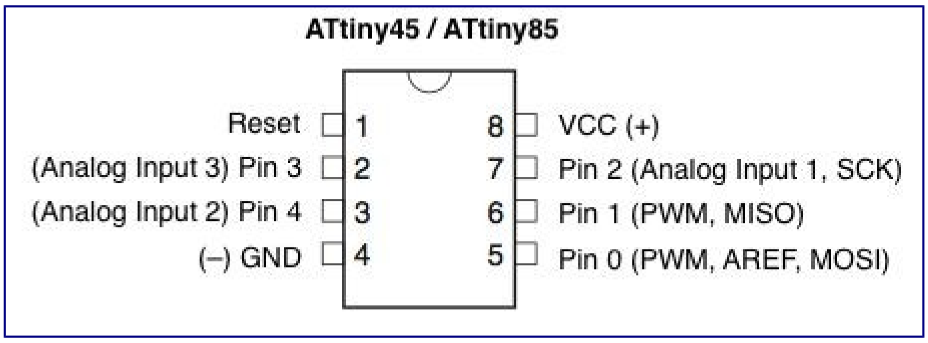
To add to the complexity pins can have more than one fuction and be labeled multiple names.

Wrapping it all together! :

Writing a program to light an LED on pin 0 on the ATtiny85 might be confusing at first because just by looking at the chip , there is no pin 0! However, by checking the datasheet of the ATtiny85 from ATMEL-snippet shown above-pin 0 is internally located on the chip's port B (and is actually the chip's *physical* pin 5)!

Step 5: Connecting the Arduino to the ATtiny Pins



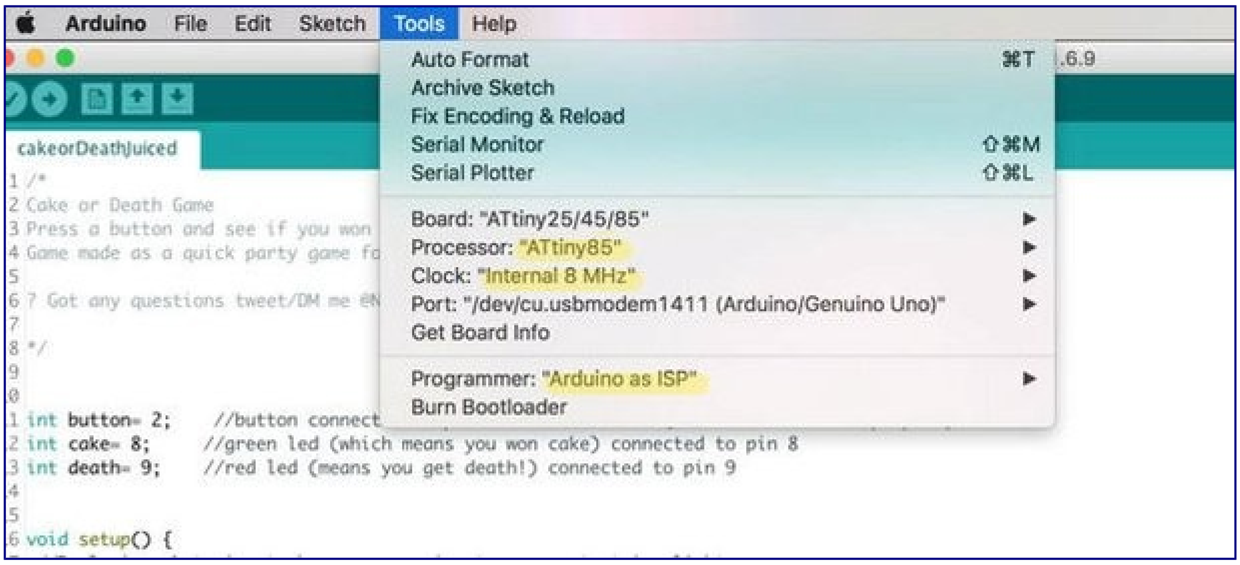


Have an electrolytic capacitor-*10uF is recommend but I used a 22uF* and it worked fine- to prevent the Arduino from restarting it's self connected to GND & RESET on the Arduino

Use a breadboard and jumper wires to make the connections bellow from the Arduino Uno to the ATtiny85:

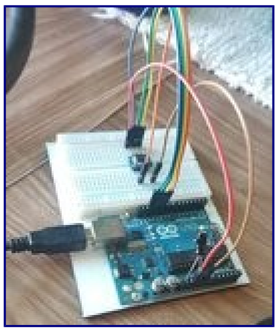
```
Arduino--> ATtiny85

5V            Vcc
GND           GND
Pin 13        Pin 2
Pin 12        Pin 1
Pin 11        Pin 0
Pin 10        Reset
```

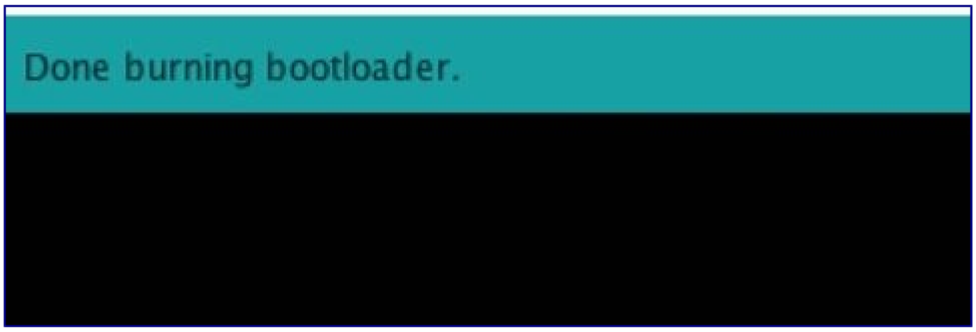
Step 6: Making the ATtiny85 Arduino Compatible



Board: "ATtiny25/45/85"
Processor: "ATtiny85"
Clock: "Internal 8 MHz"
Port: "/dev/cu.usbmodem1411 (Arduino/Genuino Uno)"
Programmer: "Arduino as ISP"

```
/*
Cake or Death Game
Press a button and see if you won
Game made as a quick party game fo

? Got any questions tweet/DM me @N

*/


int button= 2;    //button connect
int cake= 8;     //green led (which means you won cake) connected to pin 8
int death= 9;    //red led (means you get death!) connected to pin 9


void setup() {
```





Done burning bootloader.

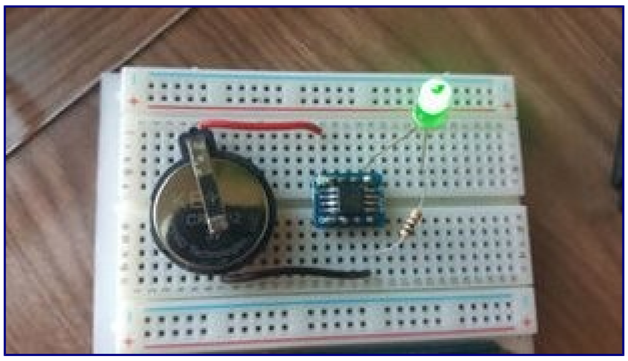
By default any fresh microcontroller chip bought will not be able to be programmed with the Arduino IDE out of the box. This is why it's required to burn the Arduino bootloader onto the chip to make sure the chip will accept any programs uploaded via the Arduino IDE.

Quick checklist before pressing "burn bootloader"

- Go to Tools -> Board scroll to the bottom select ATtiny25/45/85
- Under Tools -> Processor--> 8 MHz (internal)
- Under Tools-->Programmer-->Arduino as ISP
- Check that all wiring, capacitor, and board selections are correct
- Finally select *Burn Bootloader*
- leave the wires connected they will be used in the next step

A message will appear saying "Done Burning Bootloader"

Step 7: Uploading the blink sketch



Test that the ATtiny85 can now receive sketches from the Arduino IDE by uploading the blink example

- Go to File-->Example-->01.Basics-->blink
- Edit the sketch by replacing pin 13 with 0
- Make sure to still have the ATtiny85 board settings from the previous step selected
- Make sure all wiring is the same as the previous step
- Upload the sketch
- Wire an LED by connecting the anode to pin 0 (physical pin 5) and the cathode to a 1K resistor connected to ground (physical pin 4)
- While a resistor is not needed since the battery provides 3v (not enough to blow up an LED) it is recommended to lower the brightness of the LED

Any request for future tutorials all welcomed! Just leave a comment bellow