

## Table of Contents

Burning the bootloader.....	2
Atmega328P with bootloader on a breadboard .....	8
Programming your Atmega328P standalone.....	9
3 <sup>rd</sup> way: Using an FTDI breakout. ....	10

The purpose of this tutorial is to help you use your Atmega328P as a standalone, without the need of an Arduino and with the use of the less needed components to boot it up and run. This will also help you save money on buying a while Arduino board, it will help you reduce the power consumption of a 'arduino', it will also help you save space of the product-project.

You will need to use the Arduino IDE for this tutorial, and some of the steps require the use of Arduino board.

Tutorial created: 05/Aug/2018

last edit: 4/3/2019

By: Christianidis Vasileios

Contact info: basilisvirus@hotmail.com

# Burning the bootloader

You just got yourself a new atmega328P or you want to separate your atmega328P from your Arduino, making a “Arduino on a breadboard” project.

In case you didn't know, Arduino is called just the platform that hosts the microcontroller, (the Atmega328P is the microcontroller).

## ATmega328 Pinout

### Arduino Pins

RESET  
Digital pin 0 (RX)  
Digital pin 1 (TX)  
Digital pin 2  
Digital pin 3 (PWM)  
Digital pin 4  
Voltage (VCC)  
Ground  
Crystal  
Crystal  
Digital pin 5  
Digital pin 6  
Digital pin 7  
Digital pin 8

Pin # 1: PC6  
Pin # 2: PD0  
Pin # 3: PD1  
Pin # 4: PD2  
Pin # 5: PD3  
Pin # 6: PD4  
Pin # 7: VCC  
Pin # 8: GND  
Pin # 9: PB6  
Pin # 10: PB7  
Pin # 11: PD5  
Pin # 12: PD6  
Pin # 13: PD7  
Pin # 14: PB0

ATmega328

Pin # 28: PC5  
Pin # 27: PC4  
Pin # 26: PC3  
Pin # 25: PC2  
Pin # 24: PC1  
Pin # 23: PC0  
Pin # 22: GND  
Pin # 21: Aref  
Pin # 20: AVCC  
Pin # 19: PB5  
Pin # 18: PB4  
Pin # 17: PB3  
Pin # 16: PB2  
Pin # 15: PB1

### Arduino Pins

Analog Input 5  
Analog Input 4  
Analog Input 3  
Analog Input 2  
Analog Input 1  
Analog Input 0  
Ground (GND)  
Analog Reference  
Voltage (VCC)  
Digital Pin 13  
Digital Pin 12  
Digital Pin 11 (PWM)  
Digital Pin 10 (PWM)  
Digital Pin 9 (PWM)

www.TheEngineeringProjects.com

You can see on the left here the pins matching of the atmega and the Arduino.

What is the purpose of the Arduino then? it makes the microcontroller more costly, since you can get a Atmega328P from just 3-4 euros and a Arduino will certainly cost more since it has the atmega embed inside it.

Well you just need the Arduino for a couple of reasons:

- 1) it protects your atmega in case you plug in a higher voltage than you should.
- 2) It makes it easier to program it since you just plug it in your computer, and its easier to

plug it on a computer.

3) Arduino has already mapped the Atmega's pinouts to more user-friendly board.

4) Its more compact compared to the same layout if you build it on a breadboard.

5) It makes it easier to promote the product (Atmega and the Arduino), so if you just google

atmega328p

All Images Videos News Books More

About 2,940,000 results (0.54 seconds)

arduino

All Images Videos News Maps More

About 90,100,000 results (0.47 seconds)

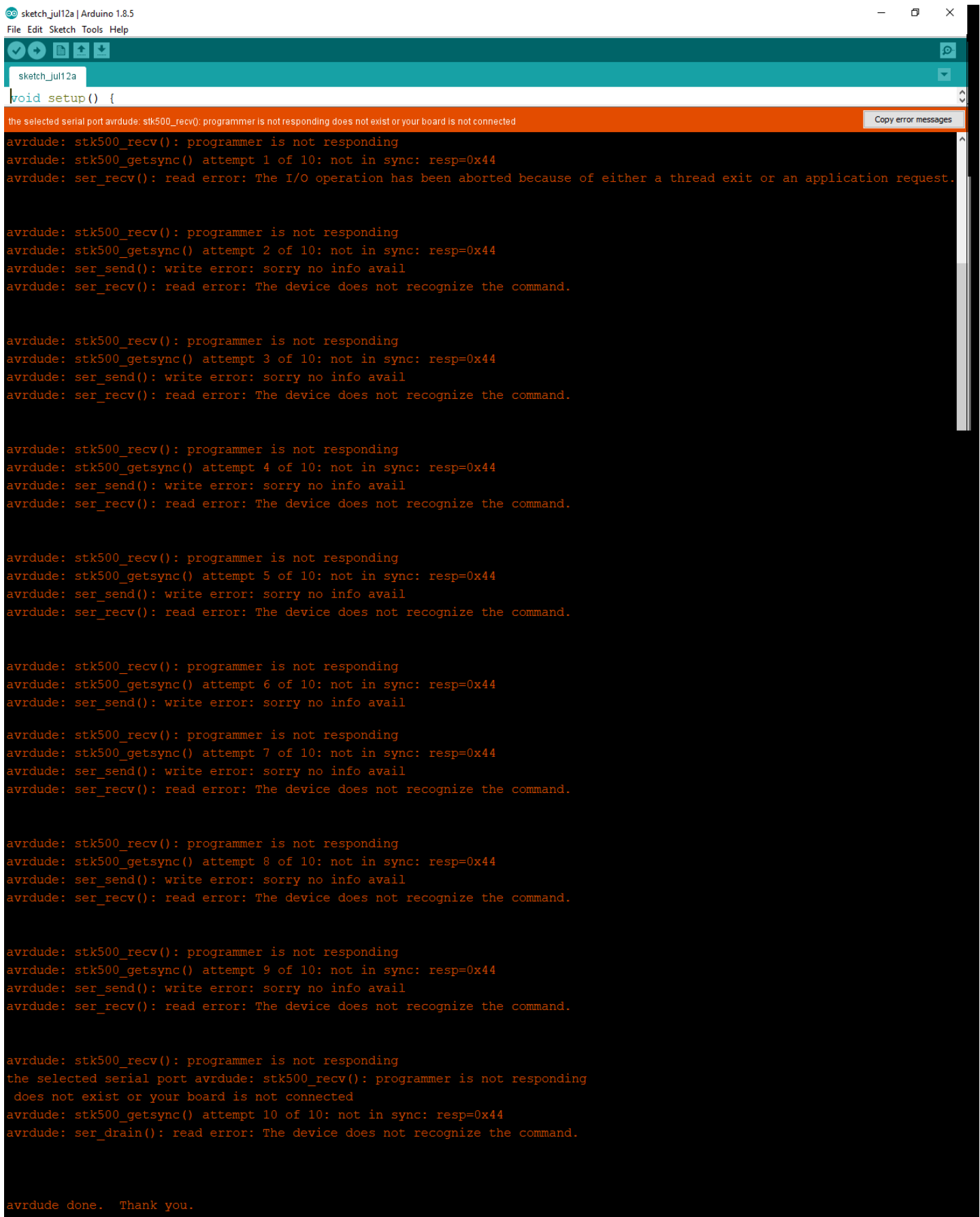
as you see there are 31 times more posts about Arduino and less about atmega, so chances are some people that use Arduino don't even know that they are actually using an Atmega, or that they can use their atmega as standalone, which I will show you how to right now.

The reasons to use Atmega as a standalone are obvious. If you put it on a pcb, it will take less space that the Arduino, it will use less energy to run, its cheaper, you can sell it to the market.

So first of, when you get a new atmega, in order to program it using Arduino IDE is to burn a bootloader on it. You see, when you want to program a microcontroller, you usually do it using a programmer, UNLESS you have a piece of firmware in your microcontroller that allows installing new firmware without the need os a programmer. This is called a bootloader.

Now, its easy to check if your atmega has a bootloader on it. You simply put your atmega on your Arduino shield, and you try to upload to it any code, even a empty one (with only setup() and loop()). If it has a bootloader, it will upload the program correctly. otherwise, if DOESN'T have a bootloader, this is the error you wil get:

Note: you can always check the official page for a couple more info: <https://www.arduino.cc/en/Hacking/Bootloader>



```
sketch_jul12a | Arduino 1.8.5
File Edit Sketch Tools Help

void setup() {

the selected serial port avrdude: stk500_rcv0: programmer is not responding does not exist or your board is not connected
Copy error messages

avrdude: stk500_rcv(): programmer is not responding
avrdude: stk500_getsync() attempt 1 of 10: not in sync: resp=0x44
avrdude: ser_rcv(): read error: The I/O operation has been aborted because of either a thread exit or an application request.

avrdude: stk500_rcv(): programmer is not responding
avrdude: stk500_getsync() attempt 2 of 10: not in sync: resp=0x44
avrdude: ser_send(): write error: sorry no info avail
avrdude: ser_rcv(): read error: The device does not recognize the command.

avrdude: stk500_rcv(): programmer is not responding
avrdude: stk500_getsync() attempt 3 of 10: not in sync: resp=0x44
avrdude: ser_send(): write error: sorry no info avail
avrdude: ser_rcv(): read error: The device does not recognize the command.

avrdude: stk500_rcv(): programmer is not responding
avrdude: stk500_getsync() attempt 4 of 10: not in sync: resp=0x44
avrdude: ser_send(): write error: sorry no info avail
avrdude: ser_rcv(): read error: The device does not recognize the command.

avrdude: stk500_rcv(): programmer is not responding
avrdude: stk500_getsync() attempt 5 of 10: not in sync: resp=0x44
avrdude: ser_send(): write error: sorry no info avail
avrdude: ser_rcv(): read error: The device does not recognize the command.

avrdude: stk500_rcv(): programmer is not responding
avrdude: stk500_getsync() attempt 6 of 10: not in sync: resp=0x44
avrdude: ser_send(): write error: sorry no info avail

avrdude: stk500_rcv(): programmer is not responding
avrdude: stk500_getsync() attempt 7 of 10: not in sync: resp=0x44
avrdude: ser_send(): write error: sorry no info avail
avrdude: ser_rcv(): read error: The device does not recognize the command.

avrdude: stk500_rcv(): programmer is not responding
avrdude: stk500_getsync() attempt 8 of 10: not in sync: resp=0x44
avrdude: ser_send(): write error: sorry no info avail
avrdude: ser_rcv(): read error: The device does not recognize the command.

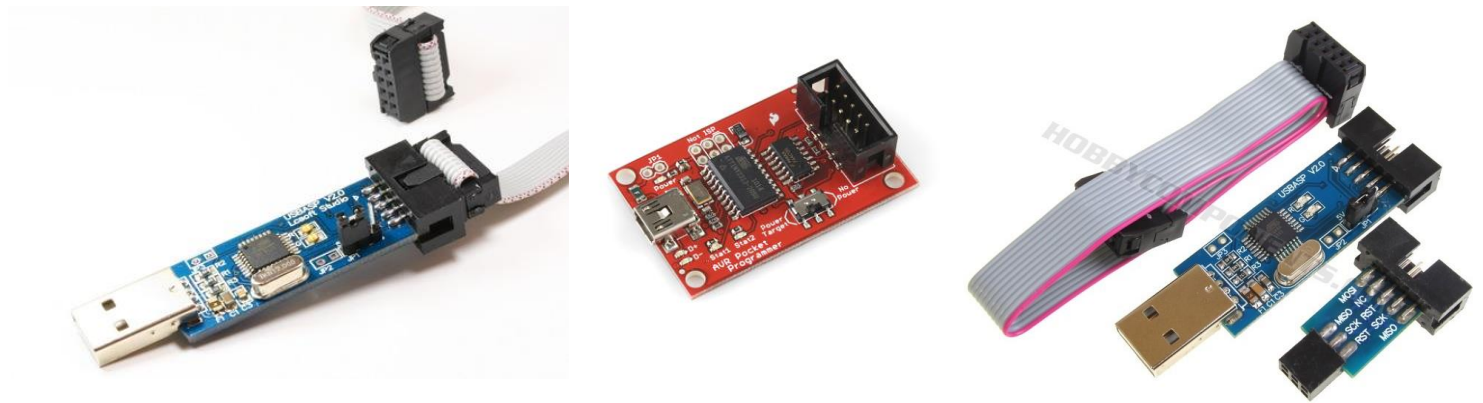
avrdude: stk500_rcv(): programmer is not responding
avrdude: stk500_getsync() attempt 9 of 10: not in sync: resp=0x44
avrdude: ser_send(): write error: sorry no info avail
avrdude: ser_rcv(): read error: The device does not recognize the command.

avrdude: stk500_rcv(): programmer is not responding
the selected serial port avrdude: stk500_rcv0: programmer is not responding
does not exist or your board is not connected
avrdude: stk500_getsync() attempt 10 of 10: not in sync: resp=0x44
avrdude: ser_drain(): read error: The device does not recognize the command.

avrdude done. Thank you.
```

Programmer not responding and not in sync. This means you don't have bootloader installed, so you cannot upload any program using Arduino ide (but you can upload a program using a programmer, I wont show how to use a programmer for now).

I will just show you how an **AVR programmer** looks like.



If you google it or look it up on an online shop, you will see that you will find many variations of this programmer. Some programmers and some avr programmers, have a microprocessor/ microcontroller on them. And this microcontroller these programmers have, is preloaded with a sketch, that works as a programmer. So, its like you are using a microcontroller to program another microcontroller. Awesome!. Just a reminder that the first programmers or the 'actual' programmers used (or use) a specific circuit that translates the data your computer is sending to the device to be programmed, accordingly.

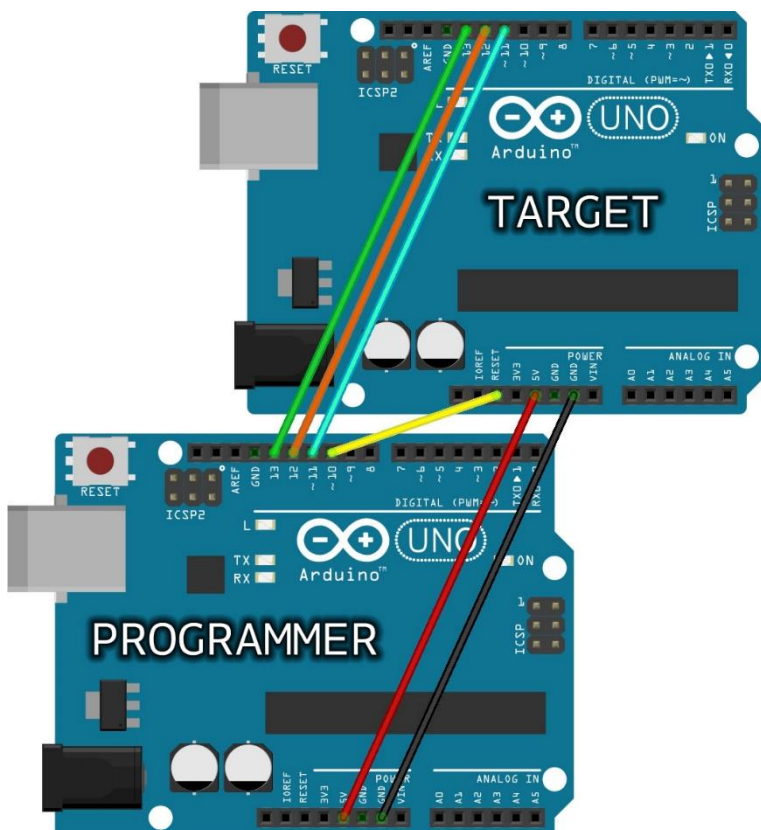
So, one way to burn a bootloader to your atmega, is to put your atmega that doesn't have a bootloader on an Arduino board, and using a second Arduino board (with a second atmega that has a bootloader on it and works just fine) as the programmer! So the second Arduino board that works just fine will be used as the programmer, but it doesn't know how does a programmer behave, it needs a sketch/code so that it knows how to burn a bootloader. We have this code ready from the arduino's examples.

And, this is how we will burn the bootloader to our new atmega:

The technique is called "**Arduino as ISP**" *In-circuit Serial Programmer (ISP)* <https://www.arduino.cc/en/Tutorial/ArduinoISP>

-Make this connection, WITHOUT powering any of the boards:

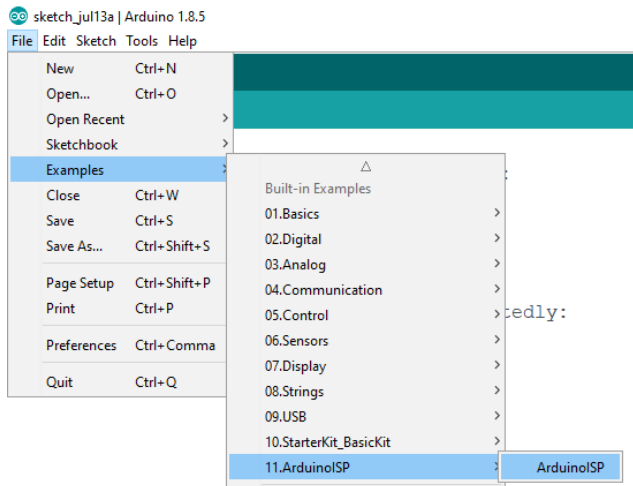
-Also on the official site it suggests: *Note: Please do not connect to USB or power supply the boards while you set up the connections and wires. We also suggest that you first program the Arduino used as ISP programmer before you proceed with the wiring to the target board.*



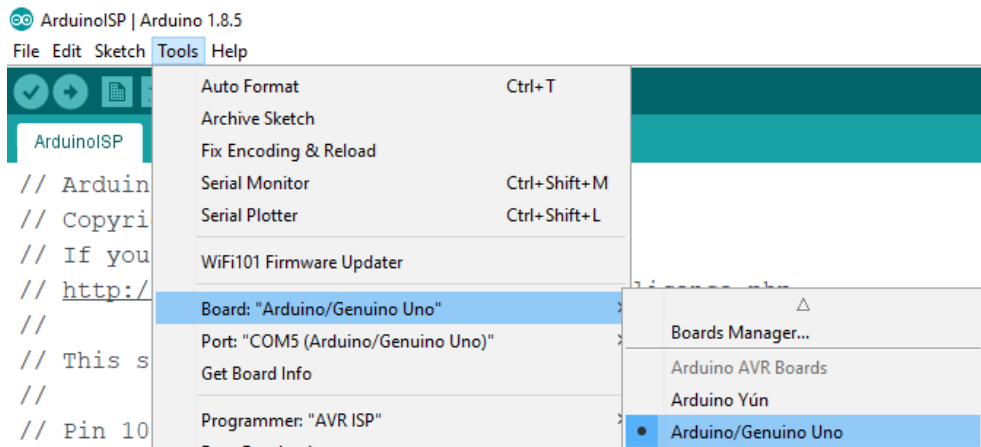
On the **target** we will put the atmega without bootloader and on the **programmer** we will put our atmega that works correctly.

You will only need to connect the **Programmer** Arduino with the ‘working-good’ atmega on it to the computer, don’t give separate power to the **target** Arduino, you will see that it will get the power that it needs from the connection we have done.

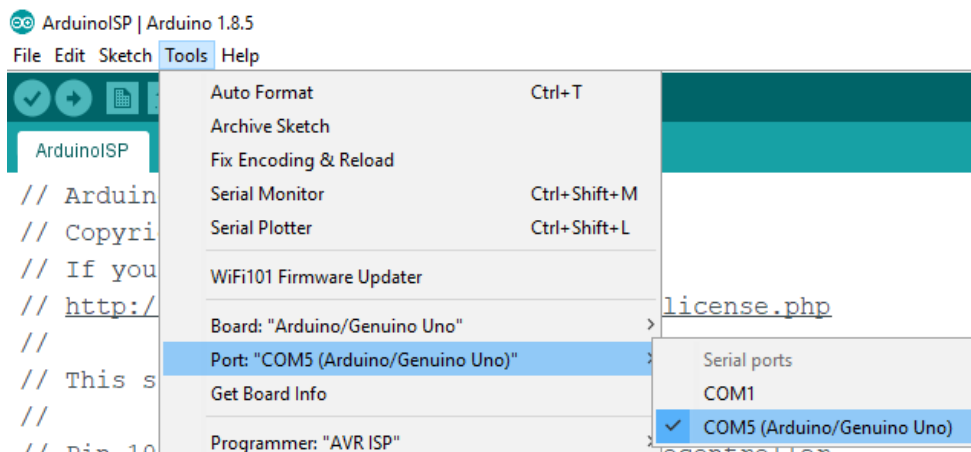
Now, select this example from the examples section:



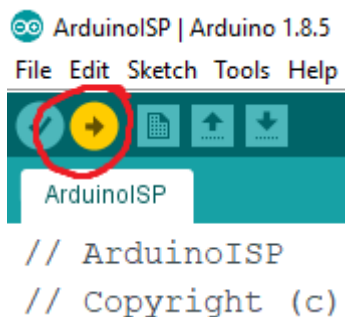
Don't forget to select the right board:



And the right COM port:



Click upload



Obviously, if you accidentally tried to upload the sketch to the wrong board (aka to the Arduino board without bootloader) you will get the error

```
Uploading...

avrdude: Version 6.3, compiled on Jan 17 2017 at 12:00:53
        Copyright (c) 2000-2005 Brian Dean, http://www.bdmicro.com/
        Copyright (c) 2007-2014 Joerg Wunsch

        System wide configuration file is "C:\Users\basil\AppData\Loc

        Using Port                : COM5
        Using Programmer           : arduino
        Overriding Baud Rate       : 115200

avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 1 of 10: not in sync: resp=0x8b
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 2 of 10: not in sync: resp=0x8b
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 3 of 10: not in sync: resp=0x8b
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 4 of 10: not in sync: resp=0x8b
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 5 of 10: not in sync: resp=0x8b
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 6 of 10: not in sync: resp=0x8b
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 7 of 10: not in sync: resp=0x8b
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 8 of 10: not in sync: resp=0x8b
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 9 of 10: not in sync: resp=0x8b
```

So if you did it correctly, it should have uploaded like any other program.

**Last step**

Now, make the connection like showed you on the pictured above, connect the board with the “Arduino as ISP” example, -which is the Arduino with the working-good atmega on it- with the computer and click “Burn bootloader”

ArduinoISP  
Copyright (c) 2008-2011 Randall Bohn  
You require a license, see  
<http://www.opensource.org/licenses/bsd-P> | Arduino 1.8.5

Tools | Help

Auto Format	Ctrl+T
Archive Sketch	
Fix Encoding & Reload	
Serial Monitor	Ctrl+Shift+M
Serial Plotter	Ctrl+Shift+L
WiFi101 / WIFININA Firmware Updater	
Board: "Arduino/Genuino Uno"	>
Port: "COM5 (Arduino/Genuino Uno)"	>
Get Board Info	
Programmer: "AVR ISP"	>
Burn Bootloader	

Wait until its completed, it will say “bootloader burned” on the Arduino ide.  
Now you can use your new atmega as regular aka putting in on a breadboard and start building the real project, or of course using it on the Arduino if you want.

Tools | Help

Auto Format	Ctrl+T
Archive Sketch	
Fix Encoding & Reload	
Manage Libraries...	Ctrl+Shift+I
Serial Monitor	Ctrl+Shift+M
Serial Plotter	Ctrl+Shift+L
WiFi101 / WIFININA Firmware Updater	
Board: "Arduino/Genuino Uno"	>
Port	>
Get Board Info	
Programmer: "Arduino as ISP"	>
Burn Bootloader	

AVR ISP  
AVRISP mkII  
USBtinyISP  
ArduinoISP  
ArduinoISP.org  
USBasp  
Parallel Programmer  
● Arduino as ISP  
Arduino Gemma  
BusPirate as ISP  
Atmel STK500 development board  
Atmel JTAGICE3 (ISP mode)  
Atmel JTAGICE3 (JTAG mode)  
Atmel-ICE (AVR)

```
default, the hardware SPI pin must be defined as the target. On all Arduino boards, the ICSP/SPI header:
```

```
MISO  . . . 5V (!)
SCK   . . . MOSI
      . . . GND
```

- Note that there is a chance that, if an error appears, you may need to:
- 1: Just before the **Last Step**, select from tools>Programmer>Arduino as ISP before you burn the bootloader. (Left picture)
  - 2: Switch to another OS! 2 years after I made this tutorial, I tried to burn a Atmega bootloader and it didn’t work! I tried on my desktop’s windows 10, it failed. Then I tried on my laptop’s Ubuntu linux and it failed. Last, I tried on my laptop’s Windows 10 and it worked finally.



There is a **second cool way** to program the atmega. You follow the same steps, just this time you don't use a second Arduino, you just press "burn bootloader" while your atmega-on breadboard is wired like this with the good-working atmega on the Arduino board:

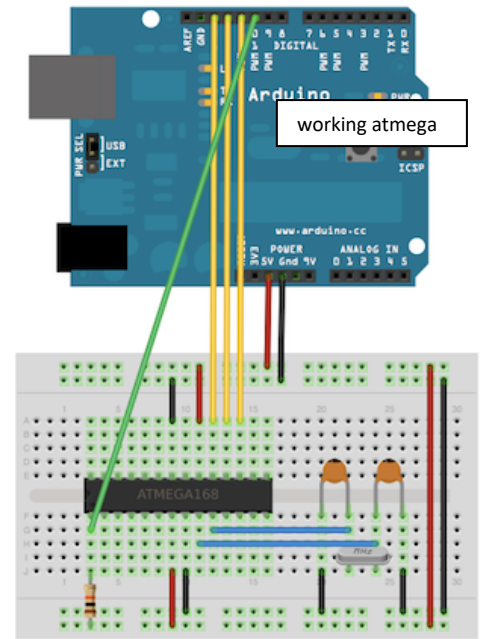
## Burning the Bootloader

If you have a new ATmega328 (or ATmega168), you'll need to burn the bootloader onto it. You can do this using an Arduino board as an in-system program (ISP). If the microcontroller already has the bootloader on it (e.g. because you took it out of an Arduino board or ordered an already-bootloaded ATmega), you can skip this section.

To burn the bootloader, follow these steps:

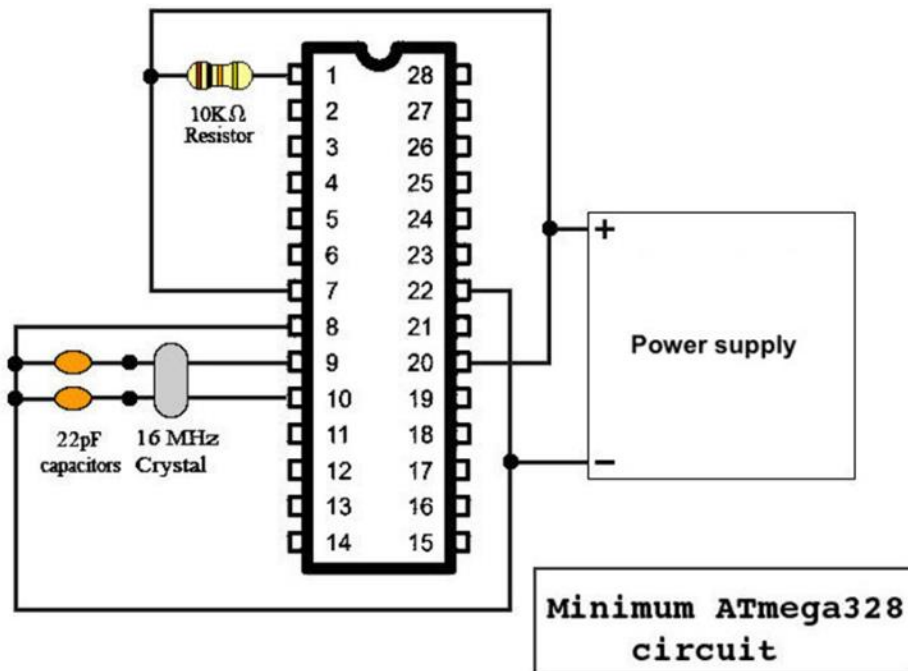
1. Upload the ArduinoISP sketch onto your Arduino board. (You'll need to select the board and serial port from the Tools menu that correspond to your board.)
2. Wire up the Arduino board and microcontroller as shown in the diagram to the right.
3. Select "Arduino Duemilanove or Nano w/ ATmega328" from the Tools > Board menu. (Or "ATmega328 on a breadboard (8 MHz internal clock)" if using the minimal configuration described below.)
4. Select "Arduino as ISP" from Tools > Programmer
5. Run Tools > Burn Bootloader

You should only need to burn the bootloader once. After you've done so, you can remove the jumper wires connected to pins 10, 11, 12, and 13 of the Arduino board.



*Using an Arduino board to burn the bootloader onto an ATmega on a breadboard.*

## Atmega328P with bootloader on a breadboard



These are the minimum connections you need to establish in order to make the atmega work properly. The power supply is 5Volts. (no more than 5.5V, you may burn the atmega)

Its not necessary to add a switch to the circuit.

The capacitor (10uF between the 5v and Gnd) is also not necessary, but its use is to get fully charged and work as a really fast power-delivery battery, whenever the atmega needs more power (more current). The capacitor is capable of delivering the current that the atmega may needs instantly, while without the capacitor you might get some current spikes every time the atmega needs more power.

The 22 pF capacitors are essential.

Note that: Atmega works on these voltage ranges:

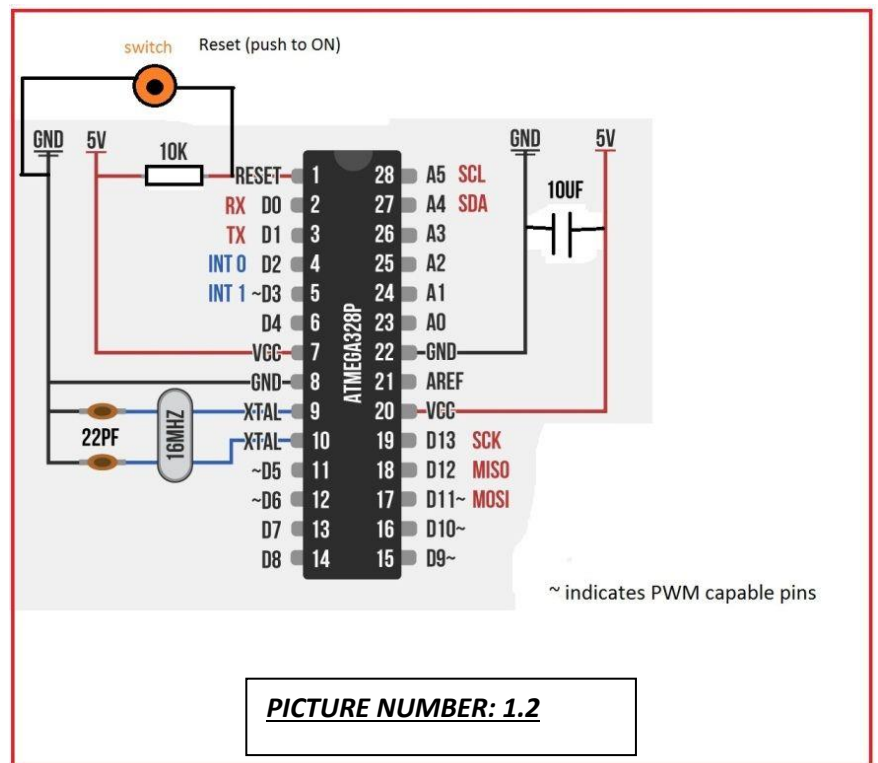
### 7.4 ATmega328P

Speed (MHz) <sup>(3)</sup>	Power Supply	Ordering Code <sup>(2)</sup>	Package <sup>(1)</sup>	Operational Range
20	1.8 - 5.5	ATmega328P- AU ATmega328P- MU ATmega328P- PU	32A 32M1-A 28P3	Industrial (-40°C to 85°C)

Commonly, 5v are used. If you need/want to use 1.8 volt you will have to lower the maximum clock speed. At 1.8V you can clock up to 4MHz, at 2.7V up to 10MHz and 4.5V up to 20MHz. You may have issues uploading your code so just feed the atmega with 4.9-5.5 volts and it will be fine, also it would be a nice idea to add the 10uF capacitor show in picture 1.2 especially if you are powering your atmega with <5V. This will help avoid power shortages.

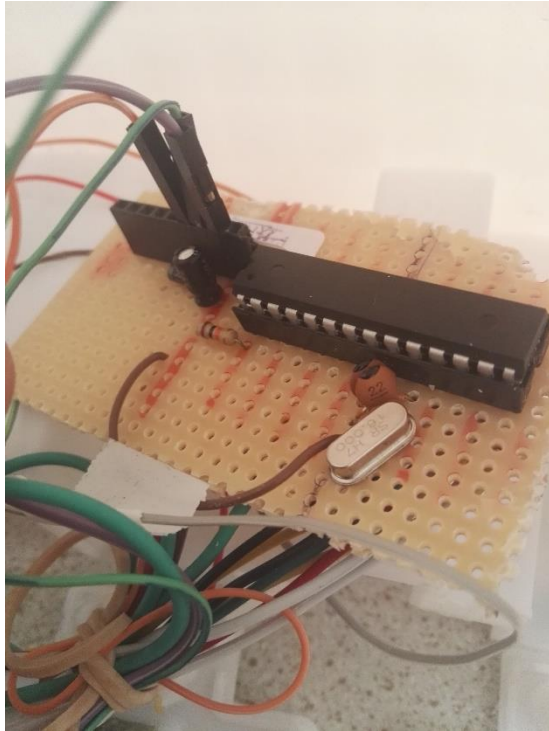
*Note: Do not connect any voltage to Aref pin of the atmega. its connected internally with its supply voltage.*

Instead of the "atmega on a breadboard" you can also say "atmega standalone".

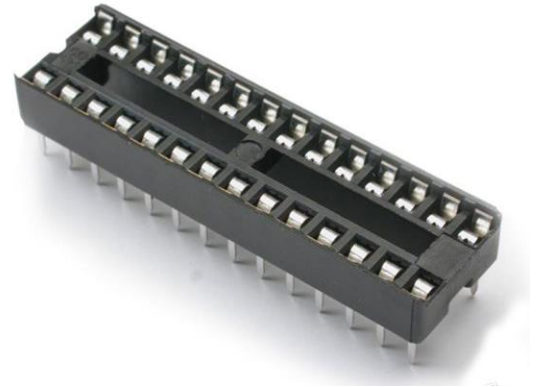




# Programming your Atmega328P standalone



After you check that your breadboard prototype works, you may want to make it more permanent, by soldering all the components on a pcb board like the picture. Take care, do not solder directly on the atmega, or you will risk of burning it. instead, do what Arduino board does. Without using the atmega, solder the Dip socket 28 pin on the pcb, then solder the necessary components on it and in the end, just plug the atmega on the Dip socket pin. That way, you wont risk damaging your Atmega due to the soldering process.



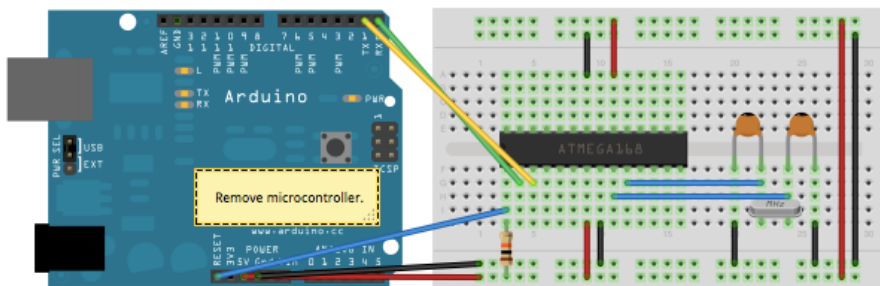
*A Dip 28-pin socket*

How to program and re-program the Atmega?

**1<sup>st</sup> way:** Put it on its Arduino board, upload the program and put it back on the breadboard/pcb.

**2<sup>nd</sup> way:** Make the Arduino board think that you have the atmega connect it on it! like so:

Connect the Tx and Rx pins on the atmega, along with the power supply, reset and ground and upload the code as usual (connect Arduino to computer and press upload).

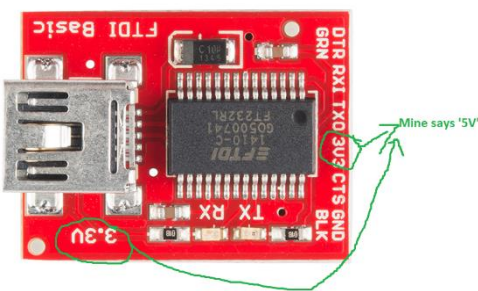
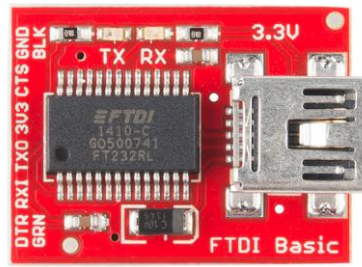
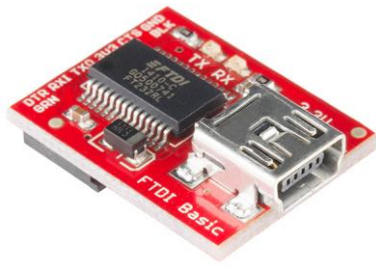


*Uploading sketches to an ATmega on a breadboard. Remember to remove the microcontroller from the Arduino board!*

### 3<sup>rd</sup> way: Using an FTDI breakout.

This is the 'professional' way to do it. If you have your Atmega328P with its bootloader burned on it, you will need to buy this:

FTDI Basic Breakout



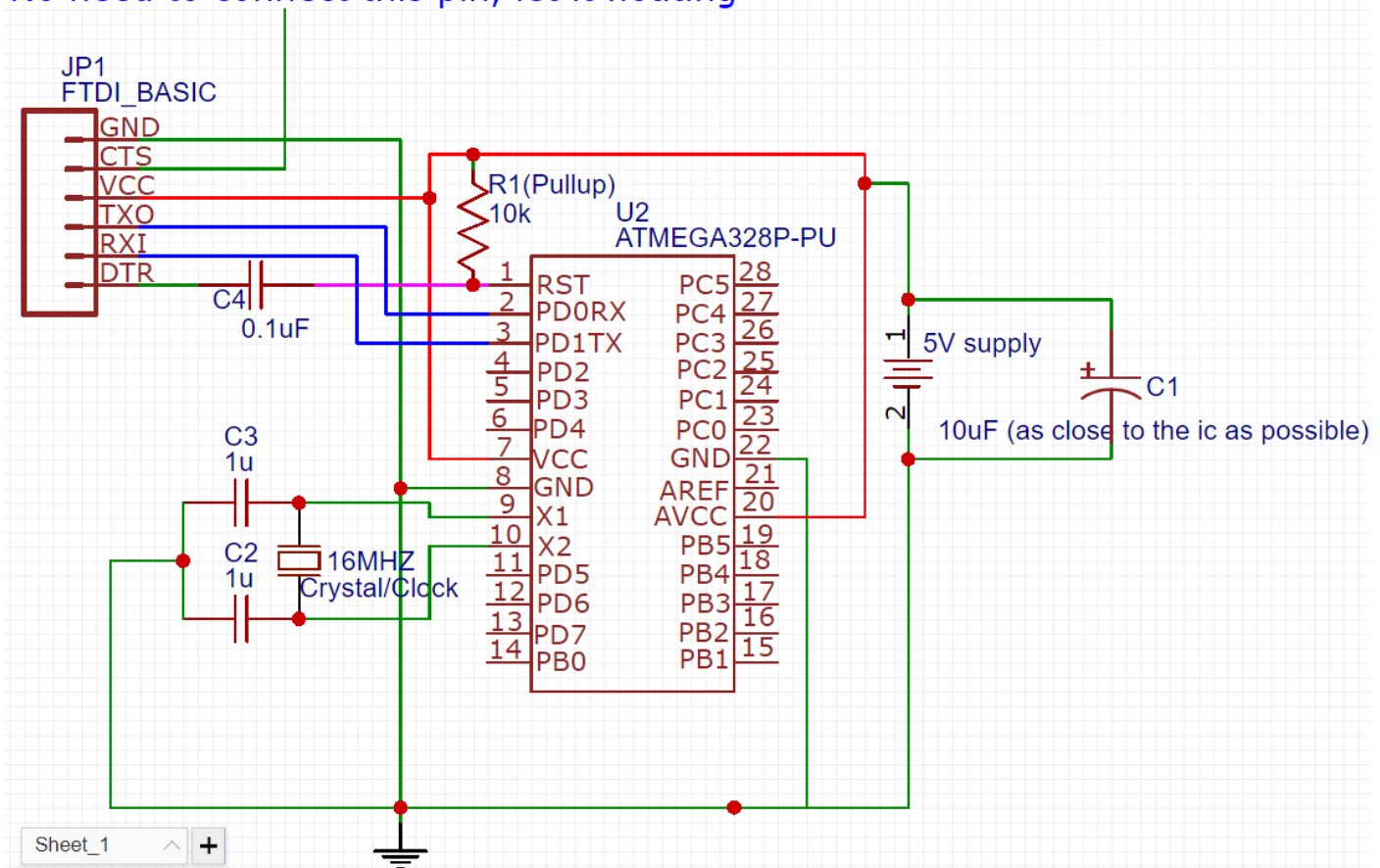
Now, here is a little difference I found out in my ftdi.

mine says '5V instead of 3V'. Make sure to use the one that says 5V, you will need it to power the Arduino.

You will need to connect your atmega with the minimum atmega328 circuit (shown in the previous page).

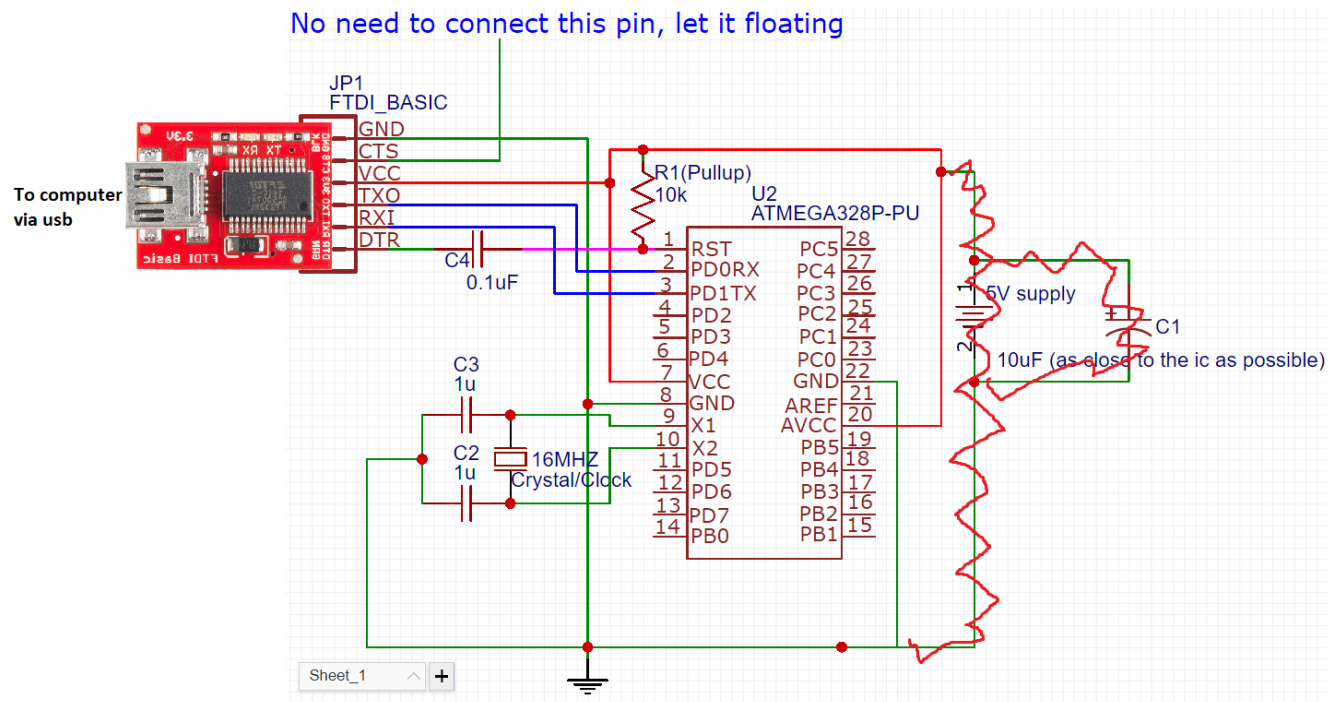
You will also need, to additionally have a 0.1uF (100nF) capacitor, and connect the circuit like so:

No need to connect this pin, let it floating

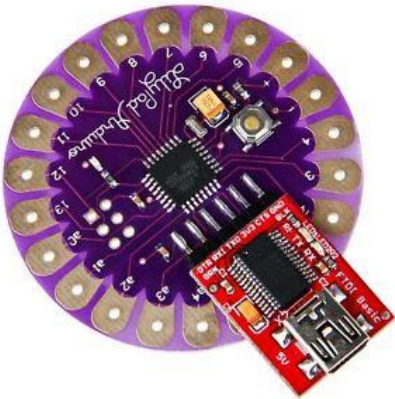


The 0.1uF capacitor is crucial, so that atmega wont reset itself repeatedly.

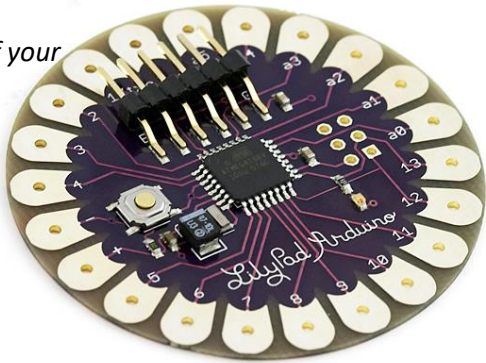
And, of course because the real-life FTDI\_BASIC also has a usb port to connect it to the computer, the actual circuit looks like this:



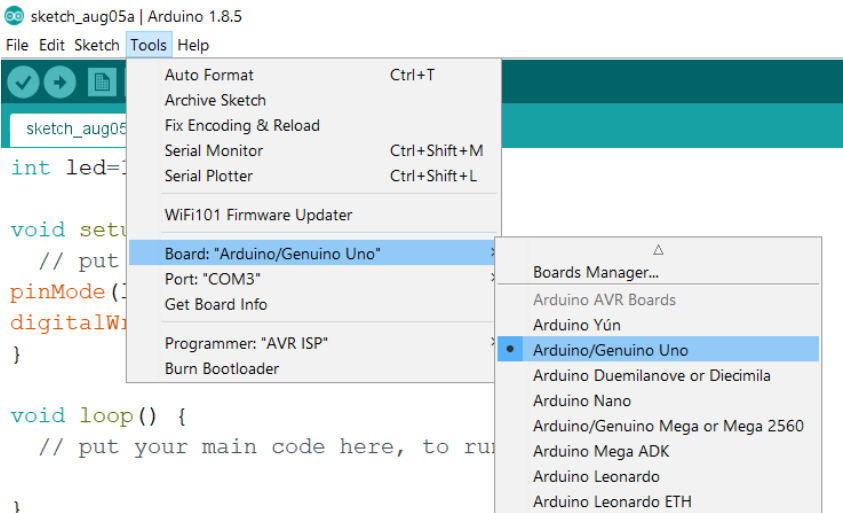
As you see, you power up the FTDI from the usb, and the FTDI powers the Atmega328P, so you don't need to power the Atmega with a external additional power (aka you don't need the 5V supply shown in the right of the above picture). You just need your Ftdi.



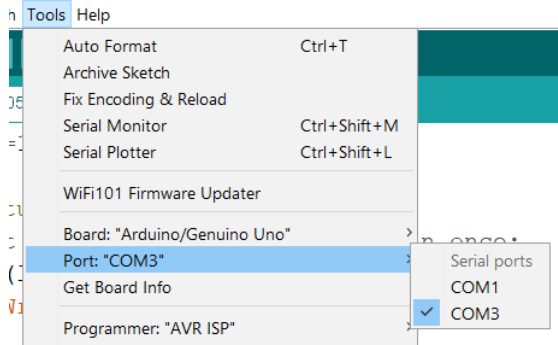
*Note: if you want to make sure that your FTDI works without any issue and is not burned, you can buy a Arduino lilypad, which has pins to connect directly to the FTDI, then connect the FTDI and the Lilypad, and use the Arduino IDE to upload any sketch to it. The lilypad is cheap (about 5 euros) and can be as well used on projects (and testing if your FTDI works), so I can recommend it to you.*



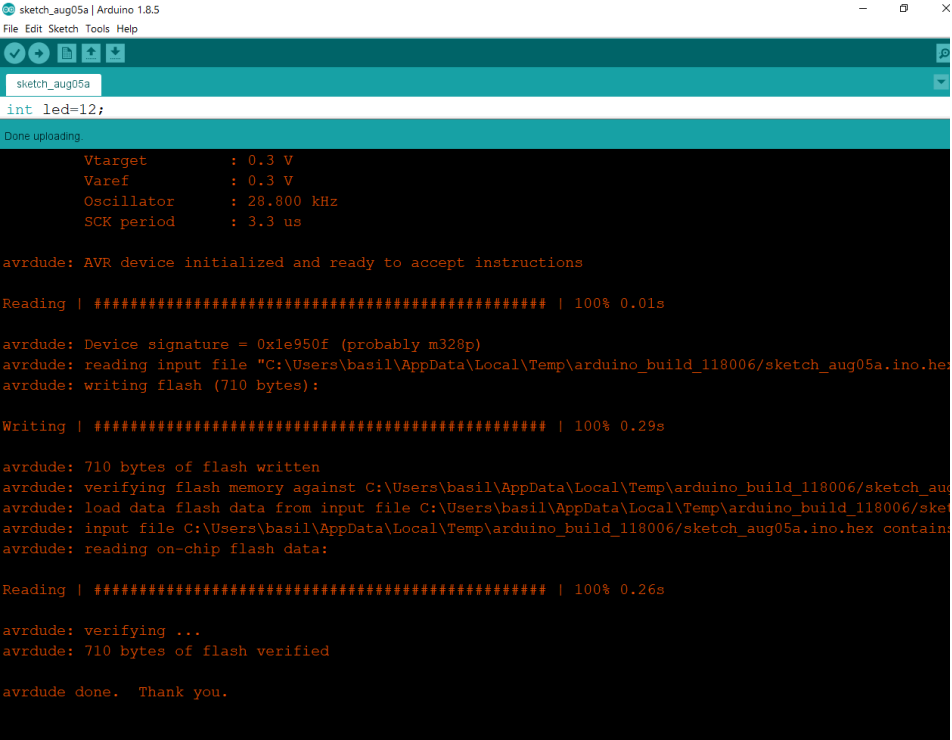
Once you have your FTDI connected on the atmega and the FTDI connected to the computer through usb, you can open the Arduino IDE, choose the board for the Arduino Uno. The ide will think that there is an arduino uno connected on it, and it will send all the code to it (to the FTDI).



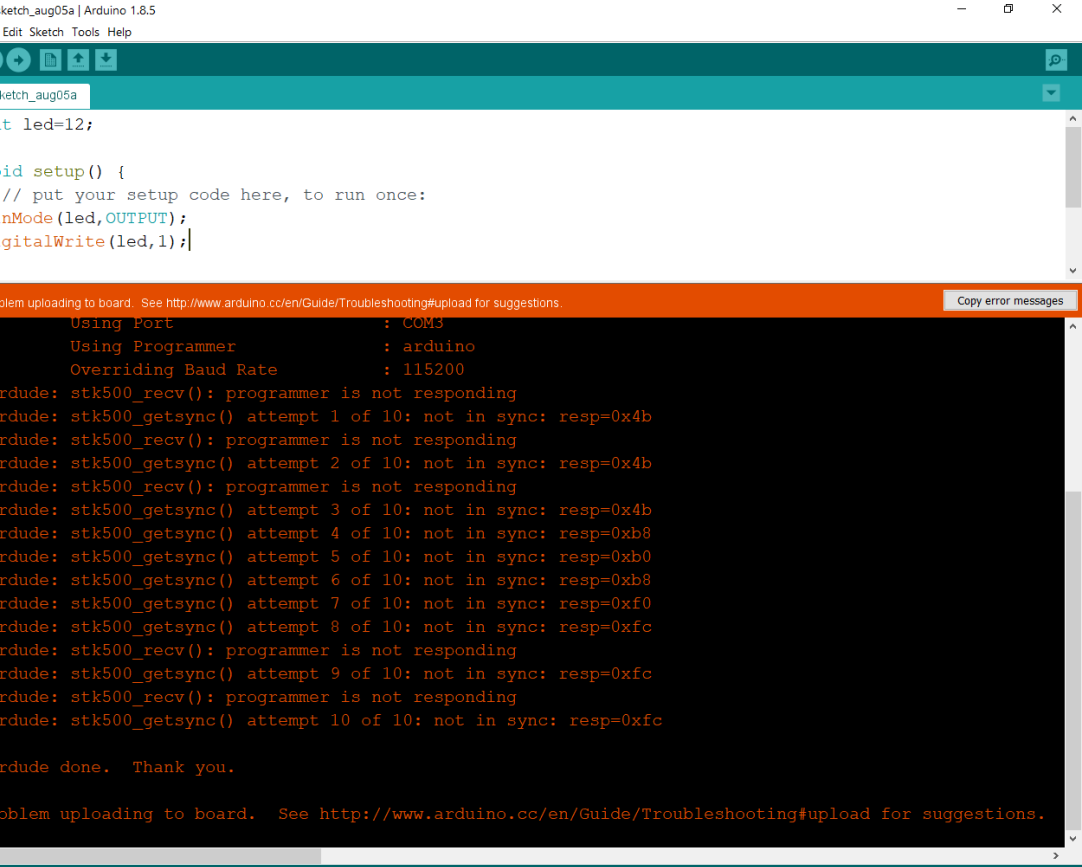
Also don't forget to pick the right port (com port). The com port wont say any name rather than the "COM3" or "COM4" etc, so you need to know which one is the right one. One way to find out is using device manager. another way is by plugging it in and out and watch which com port names change or dissapear. Once you have selected the right port, you can press upload program, it will compile and upload the sketch.



If you have done everything right it should upload correctly and you should see this:



If you have not used a 0.1uF capacitor between DTR and pin 1 of atmega OR you didn't connect the timer (crystal), it will show you this:



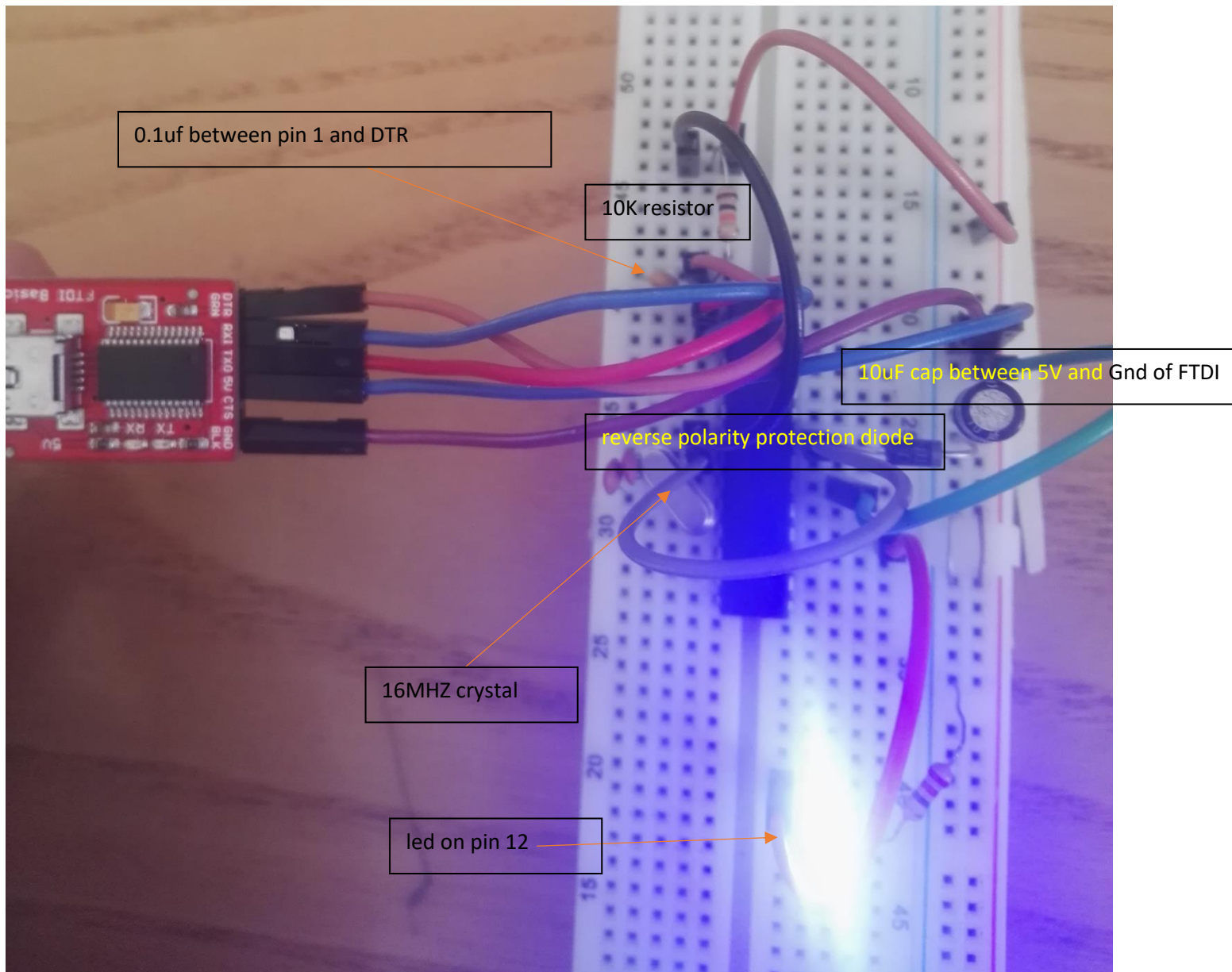
And it will also throw you an error if you don't have the crystal connected. you can also try to upload the sketch without the 0.1uF capacitor if this error occurs, and double check your circuit.



As you can see I did not upload an empty sketch, I wrote a small piece of code to light up a small led on the pin 12 of the atmega, to make sure that its working good.  
the code:

```
int led=12;
void setup() {
  // put your setup code here, to run once:
  pinMode(led,OUTPUT);
  digitalWrite(led,1);
}
void loop() {
  // put your main code here, to run repeatedly:
}
```

How mine looks like in the end:



Tutorial created: 05/Aug/2018

last edit: 4/3/2019

By: Christianidis Vasileios

Contact info: [basilisvirus@hotmail.com](mailto:basilisvirus@hotmail.com)