

Table of Contents

Burning the bootloader.....	2
Atmega328P with bootloader on a breadboard.....	7
Programming your Atmega328P standalone.....	8
3 rd way: Using an FTDI breakout.....	9
Atmega328p-pu Debugging. How things should be.....	13
1)XTAL without ceramic capacitors.....	13
2)XTAL with ceramic capacitors.....	15
Bonus: Why you should not use electrolytic capacitors, or larger than 22pF capacitors:.....	16
Programming the mcu, oscilloscope on the UART pins.....	17

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: 22/Oct/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	Pin # 1: PC6	Pin # 28:PC5	Analog Input 5
Digital pin 0 (RX)	Pin # 2: PD0	Pin # 27:PC4	Analog Input 4
Digital pin 1 (TX)	Pin # 3: PD1	Pin # 26:PC3	Analog Input 3
Digital pin 2	Pin # 4: PD2	Pin # 25:PC2	Analog Input 2
Digital pin 3 (PWM)	Pin # 5: PD3	Pin # 24:PC1	Analog Input 1
Digital pin 4	Pin # 6: PD4	Pin # 23:PC0	Analog Input 0
Voltage (VCC)	Pin # 7: VCC	Pin # 22:GND	Ground (GND)
Ground	Pin # 8: GND	Pin # 21:Aref	Analog Reference
Crystal	Pin # 9: PB6	Pin # 20:AVCC	Voltage (VCC)
Crystal	Pin # 10:PB7	Pin # 19:PB5	Digital Pin 13
Digital pin 5	Pin # 11:PD5	Pin # 18:PB4	Digital Pin 12
Digital pin 6	Pin # 12:PD6	Pin # 17:PB3	Digital Pin 11 (PWM)
Digital pin 7	Pin # 13:PD7	Pin # 16:PB2	Digital Pin 10 (PWM)
Digital pin 8	Pin # 14:PB0	Pin # 15:PB1	Digital Pin 9 (PWM)

Arduino Pins

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 makes it easier to program it since you just plug it in your computer, and its easier to plug it on a computer.
- 2) Arduino has already mapped the Atmega's

pinouts to more user-friendly board.

- 3) Its more compact compared to the same layout if you build it on a breadboard.
- 4) It makes it easier to promote the product (Atmega and the Arduino), so if you just google

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 of a programmer. This piece of firmware 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, you will get errors.

If you get errors, 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).

This is how an **AVR programmer** looks, it is called Atmel ICE.



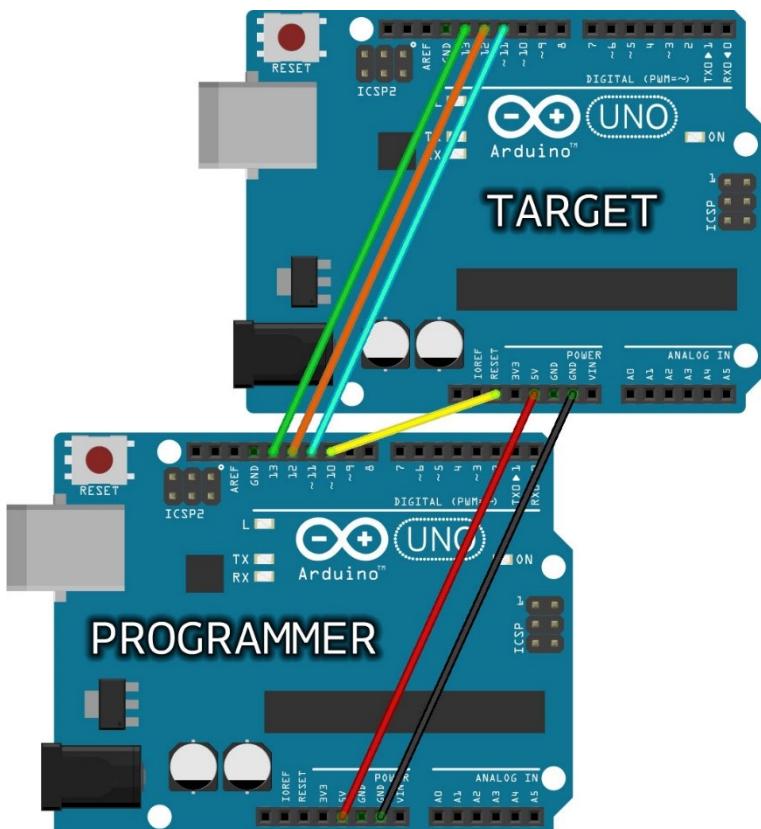
So, one way to burn a bootloader to your atmega, is to put your atmega that doesn't have a bootloader on an Arduino board, and using a second Arduino board (with a second atmega that has a bootloader on it and works just fine) as the programmer! So the second Arduino board that works just fine will be 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>

-Do not make the connections yet.

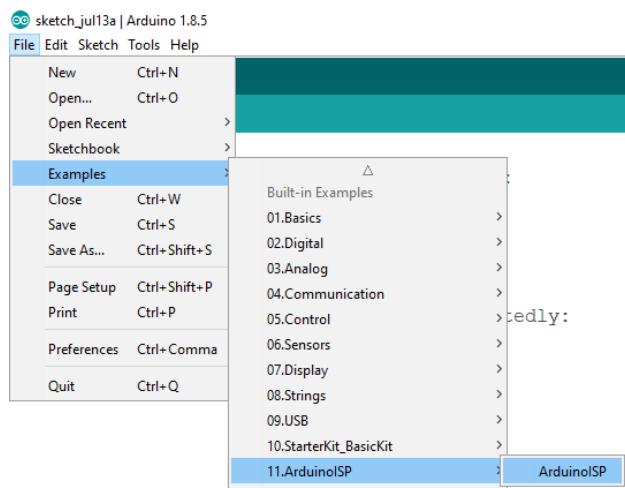
-Also on the official site it suggests: *Note: Please do not connect to USB or power supply the boards while you set up the connections and wires. We also suggest that you first program the Arduino used as ISP programmer before you proceed with the wiring to the target board.*



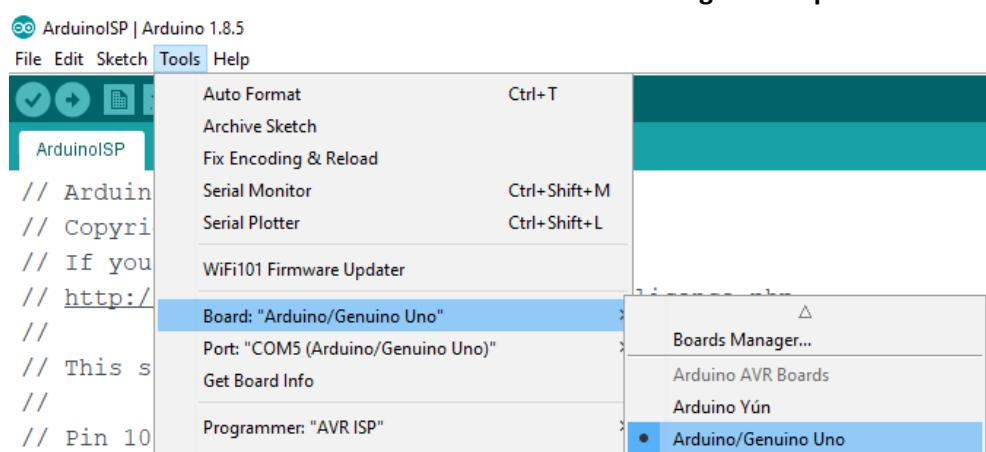
On the **target** we will put the atmega without bootloader and on the **programmer** we will put our atmega that works correctly.

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:

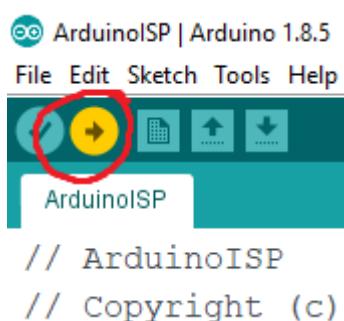


**Don't forget to select the right board
And the right COM port:**

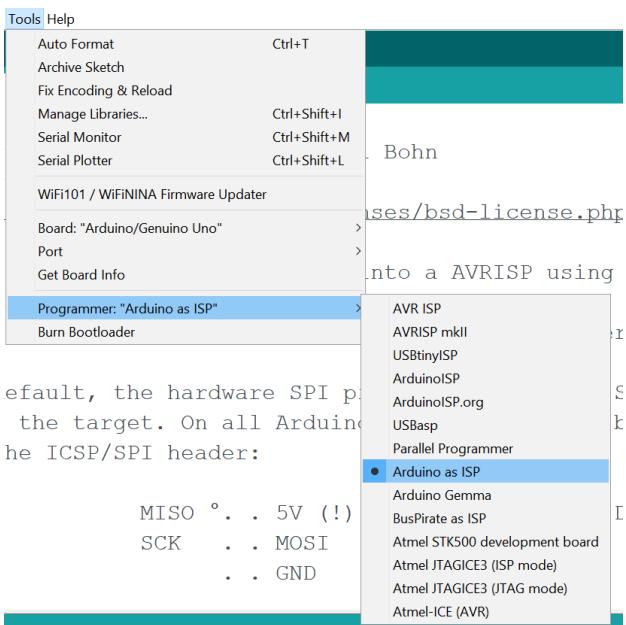


Remember, we program the normally working arduino now, so connect that only, to program it.

Click Upload



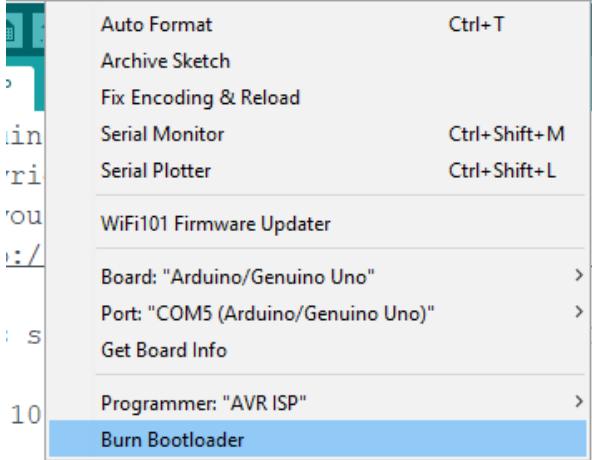
Now make the connections shown above between the two boards, and then proceed.



select from tools>Programmer>Arduino as ISP before you burn the bootloader. (Left picture)

Finally, burn the bootloader

```
inoISP
right (c) 2008-2011 Randall Bohn
you require a license, see
http://www.opensource.org/licenses/bsd-
P | Arduino 1.8.5
Sketch Tools Help
```



So if you did it correctly, it should have uploaded like any other program, but a bit slower.

Now you can use your new atmega as regular aka putting it on a breadboard and start building the real project, or of course using it on the Arduino if you want.

Note that there is a chance that, if an error appears, you may need to:

- 1: 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.
- 2: Unplug the arduino to the computer, and plug it back. This can solve a bug where the whole process would not work.
- 3: Make sure you selected Programmer = Arduino as ISP **not** ArdunolISP

There is a ‘**second’ cool way** to program the atmega, if you dont have a second arduino board. 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:

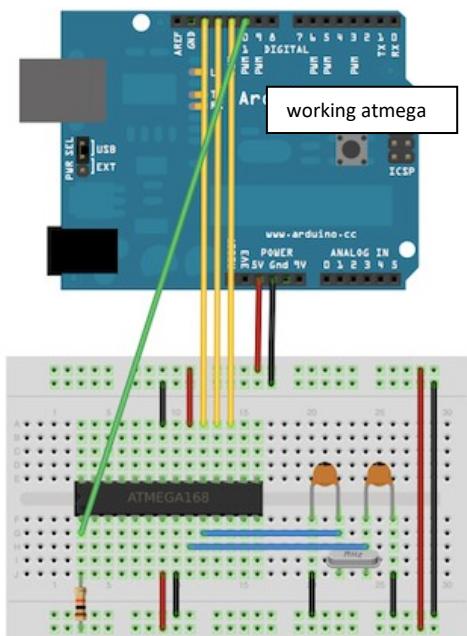
Burning the Bootloader

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:

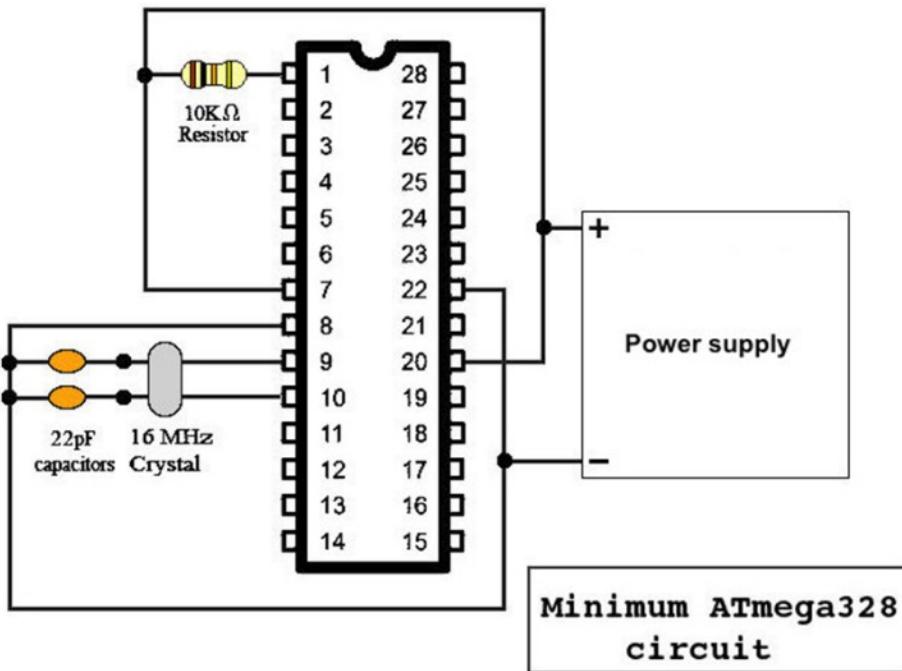
1. Upload the ArduinoISP sketch onto your Arduino board. (You'll need to select the board and serial port from the Tools menu that correspond to your board.)
2. Wire up the Arduino board and microcontroller as shown in the diagram to the right.
3. Select "Arduino Duemilanove or Nano w/ ATmega328" from 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

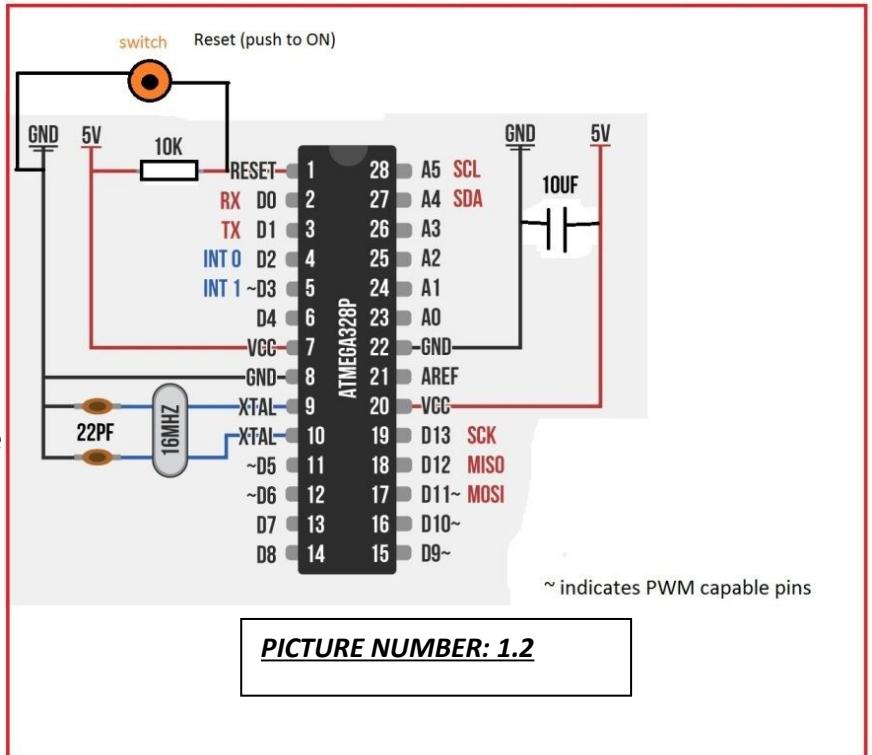


Its not necessary to add a switch to the circuit.

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 not essential. Datasheet says you can put from 12pF to 22pF there.

Note that: Atmega works on these voltage ranges:



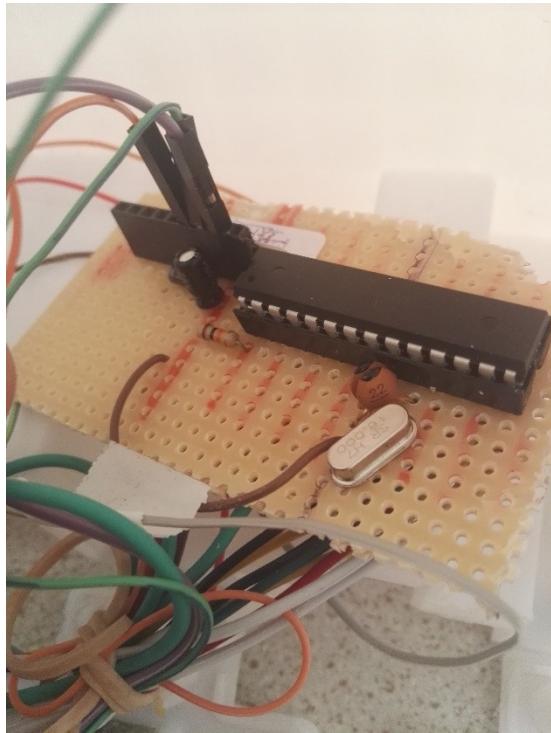
7.4 ATmega328P

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

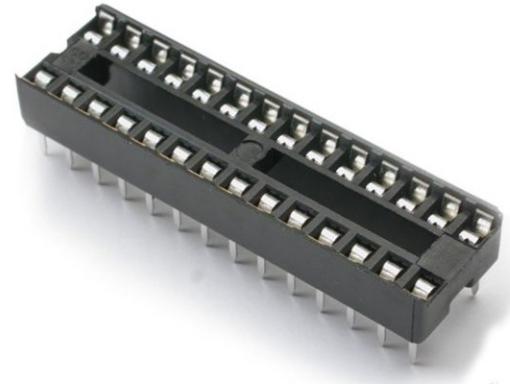
Commonly, 5v are used. If you need/want to use 1.8 volt you will have to lower the maximum clock speed. At 1.8V you can clock up to 4MHz, at 2.7V up to 10MHz and 4.5V up to 20MHz. You may have issues uploading your code so just feed the atmega with 4.9-5.5 volts and it will be fine, also it would be 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.



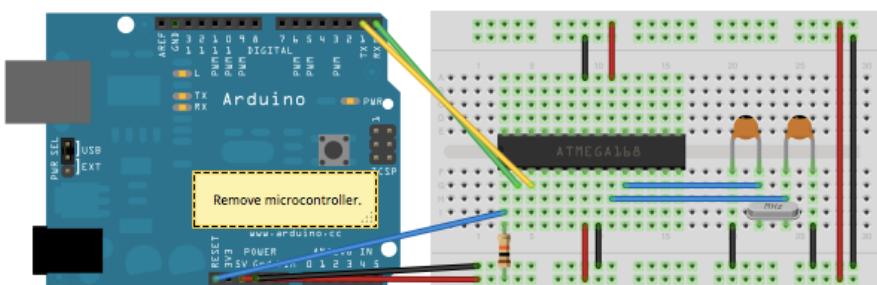
A Dip 28-pin socket

How to program and re-program the Atmega?

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

2nd 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!

3rd way: Using an FTDI breakout.

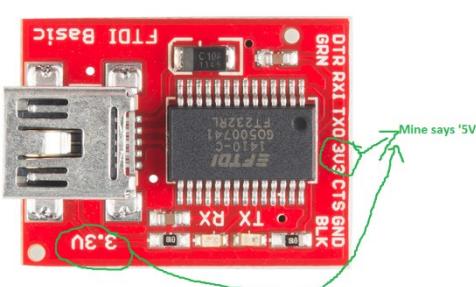
This is the 'professional' way to do it. If you have your Atmega328P with its bootloader burned on it, you will need to buy this:

FTDI Basic Breakout



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

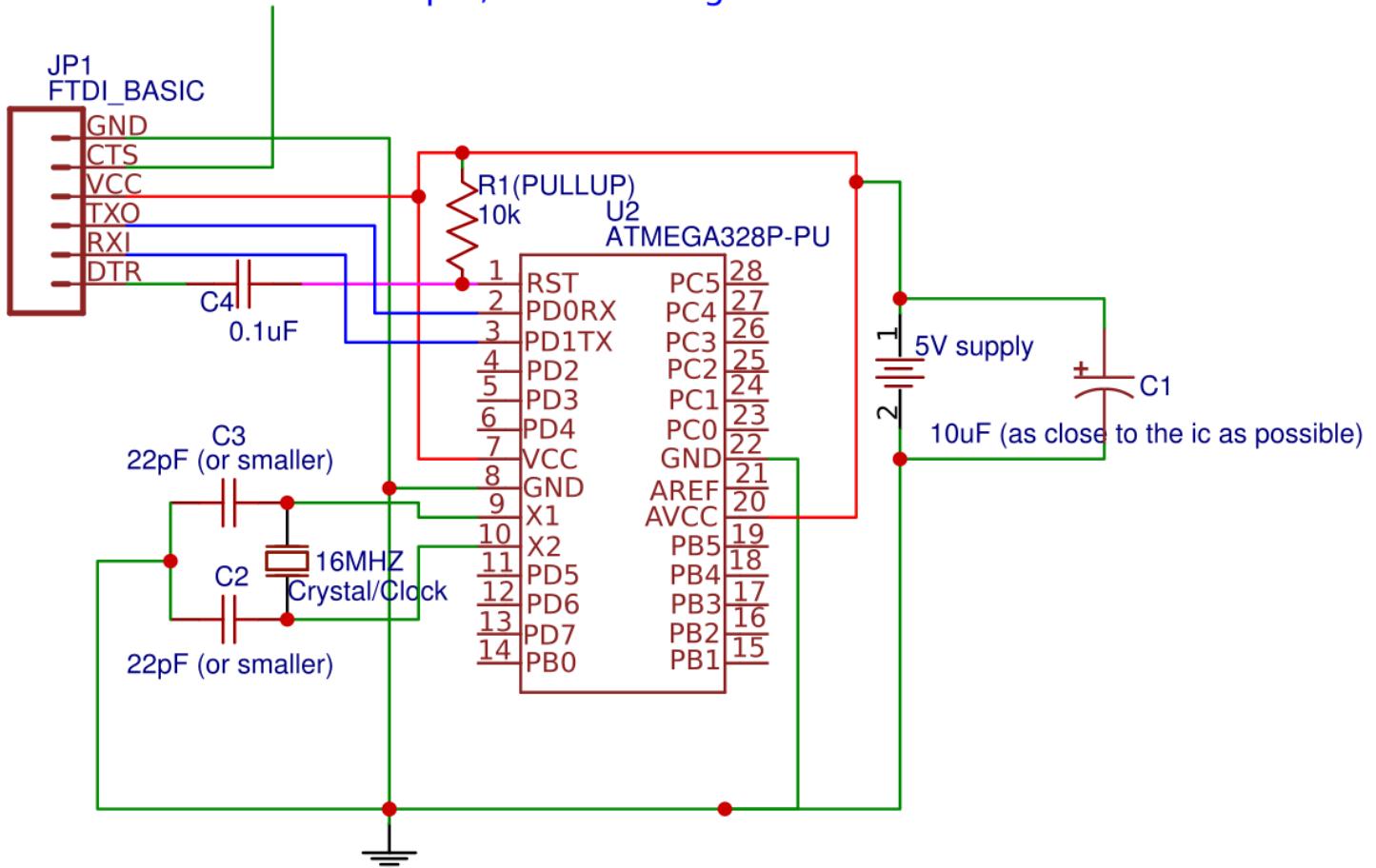
mine says '5V instead of 3v'. Make sure to use the one that says 5V, you will need it to power the Arduino.



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



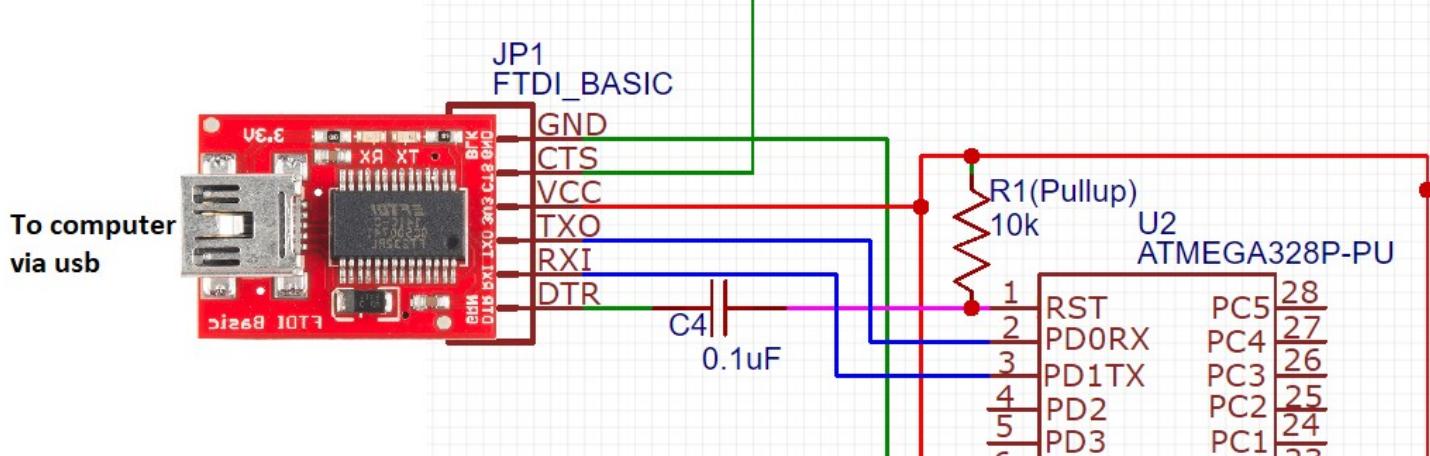
The 0.1uF capacitor is crucial, so that the programming can start.

10k is called 'pull-up resistor' and it keeps the atmega328 from resetting itself.

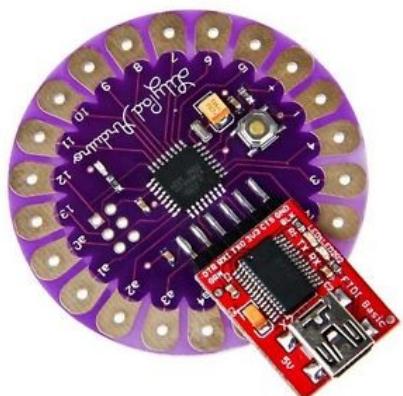
You don't need to connect the 22pF, the crystal will work good without these as well. They are good to be connected for smoothening. But take care: do not connect bigger capacitors than 22pF, because they will store the voltage used for oscillation, and the circuit will not work.

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:

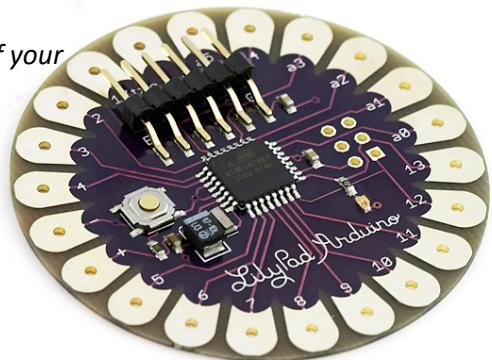
No need to connect this pin, let it floating



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.

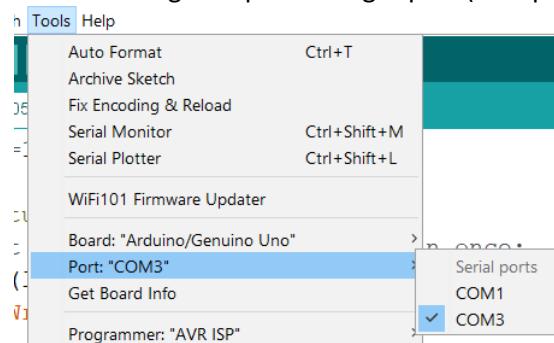


Once you have your FTDI connected on the atmega and the FTDI connected to the computer through usb, you can open

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).

```
sketch_aug05a | Arduino 1.8.5
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Arduino/Genuino Uno"
Port: "COM3"
Get Board Info
Programmer: "AVR ISP"
Burn Bootloader
int led=1;
void setup()
// put
pinMode(1,OUTPUT);
digitalWrite(1,LOW);
}
void loop()
// put your main code here, to run
```

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:

```
sketch_aug05a | Arduino 1.8.5
File Edit Sketch Tools Help
sketch_aug05a
int led=12;
Done uploading
Vtarget      : 0.3 V
Varef        : 0.3 V
Oscillator   : 28.800 kHz
SCK period   : 3.3 us

avrdude: AVR device initialized and ready to accept instructions

Reading | ##### | 100% 0.01s

avrdude: Device signature = 0x1e950f (probably m328p)
avrdude: reading input file "C:\Users\basil\AppData\Local\Temp\arduino_build_118006\sketch_aug05a.ino.hex"
avrdude: writing flash (710 bytes):

Writing | ##### | 100% 0.29s

avrdude: 710 bytes of flash written
avrdude: verifying flash memory against C:\Users\basil\AppData\Local\Temp\arduino_build_118006\sketch_aug05a.ino.hex...
avrdude: load data flash data from input file C:\Users\basil\AppData\Local\Temp\arduino_build_118006\sketch_aug05a.ino.hex
avrdude: input file C:\Users\basil\AppData\Local\Temp\arduino_build_118006\sketch_aug05a.ino.hex contains 710 bytes
avrdude: reading on-chip flash data:

Reading | ##### | 100% 0.26s

avrdude: verifying ...
avrdude: 710 bytes of flash verified

avrdude done. Thank you.
```

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:

```
sketch_aug05a | Arduino 1.8.5
File Edit Sketch Tools Help
sketch_aug05a
int led=12;

void setup() {
  // put your setup code here, to run once:
  pinMode(led,OUTPUT);
  digitalWrite(led,1);
}

Problem uploading to board. See http://www.arduino.cc/en/Guide/Troubleshooting#upload for suggestions.
Using Port          : COM3
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=0x4b
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 2 of 10: not in sync: resp=0x4b
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 3 of 10: not in sync: resp=0x4b
avrdude: stk500_getsync() attempt 4 of 10: not in sync: resp=0xb8
avrdude: stk500_getsync() attempt 5 of 10: not in sync: resp=0xb0
avrdude: stk500_getsync() attempt 6 of 10: not in sync: resp=0xb8
avrdude: stk500_getsync() attempt 7 of 10: not in sync: resp=0xf0
avrdude: stk500_getsync() attempt 8 of 10: not in sync: resp=0xfc
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 9 of 10: not in sync: resp=0xfc
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 10 of 10: not in sync: resp=0xfc

avrdude done. Thank you.

Problem uploading to board. See http://www.arduino.cc/en/Guide/Troubleshooting#upload for suggestions.
```

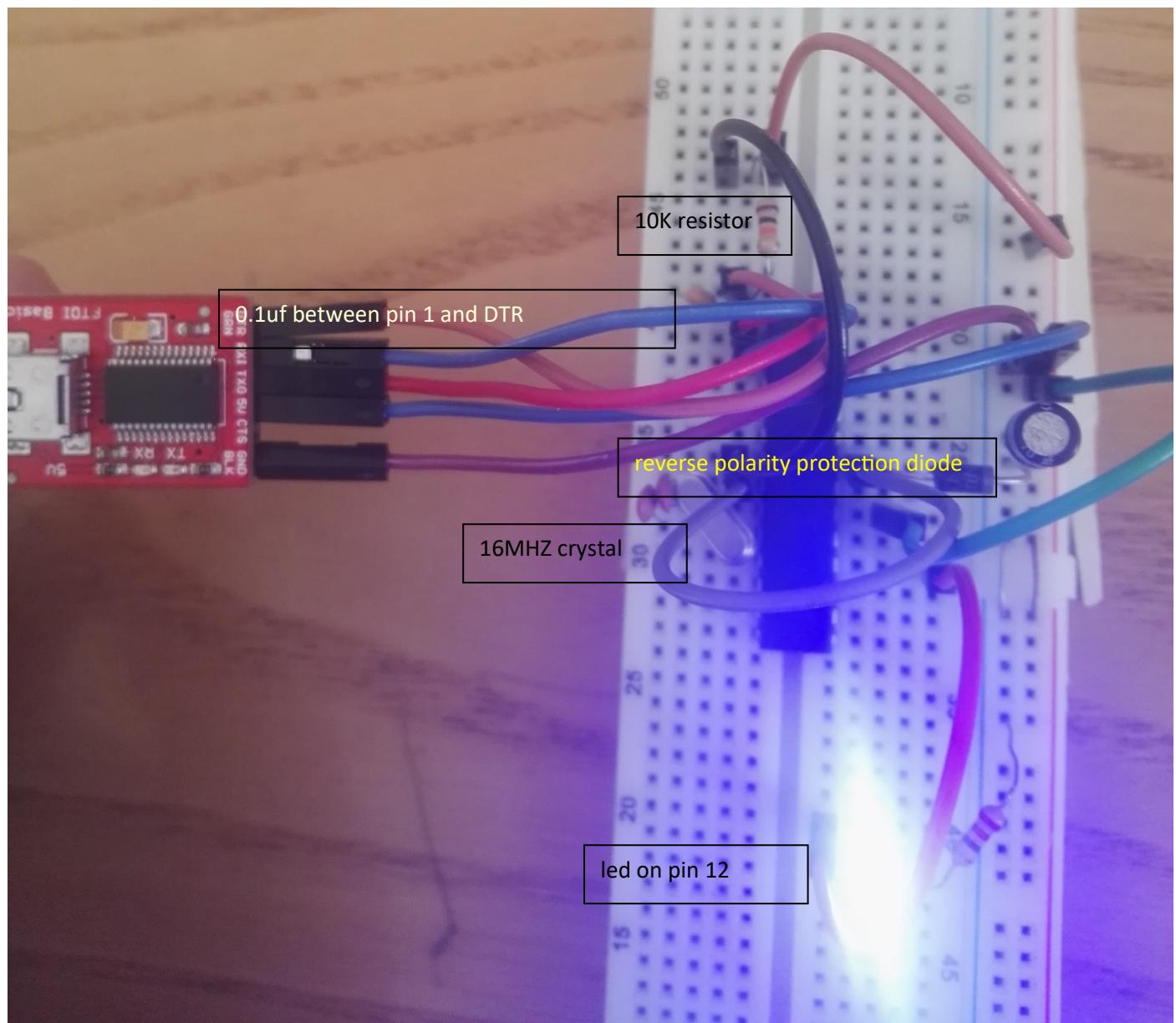
And it will also throw you an error if you don't have the crystal connected. you can also try to upload the sketch without the 0.1uF capacitor if this error occurs, and double check your circuit.

As you can see I did not uploaded 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:  
}
```

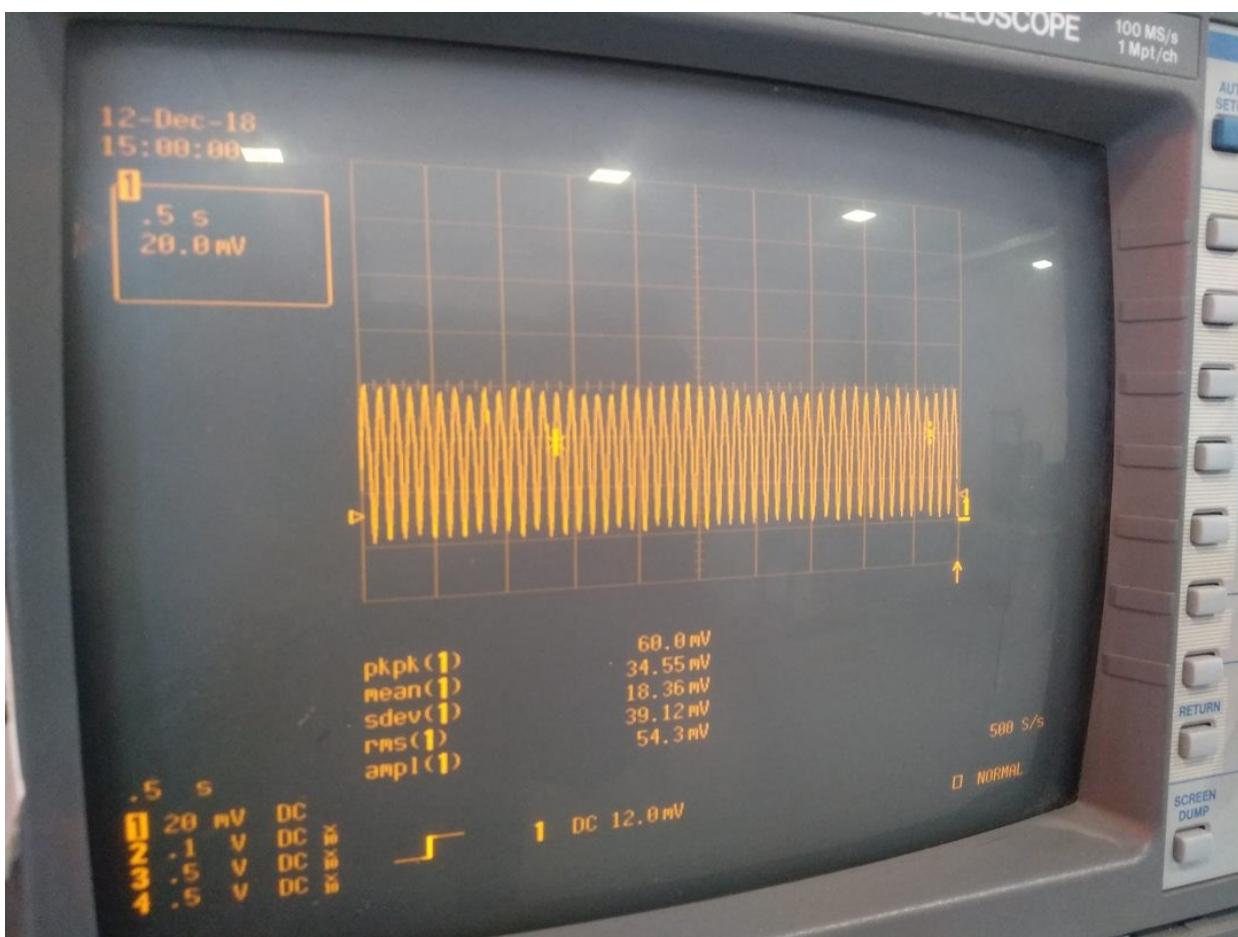
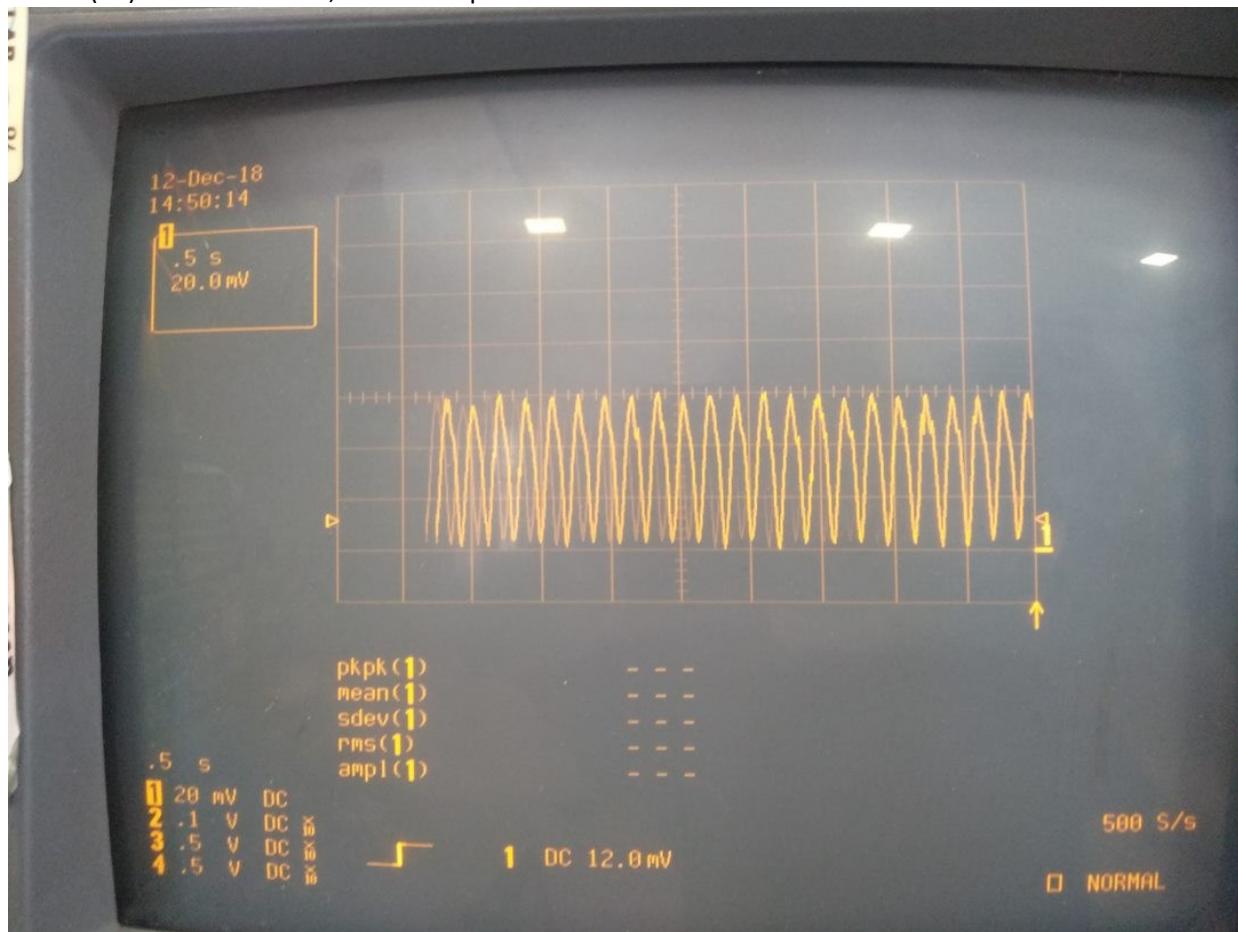
How mine looks like in the end:



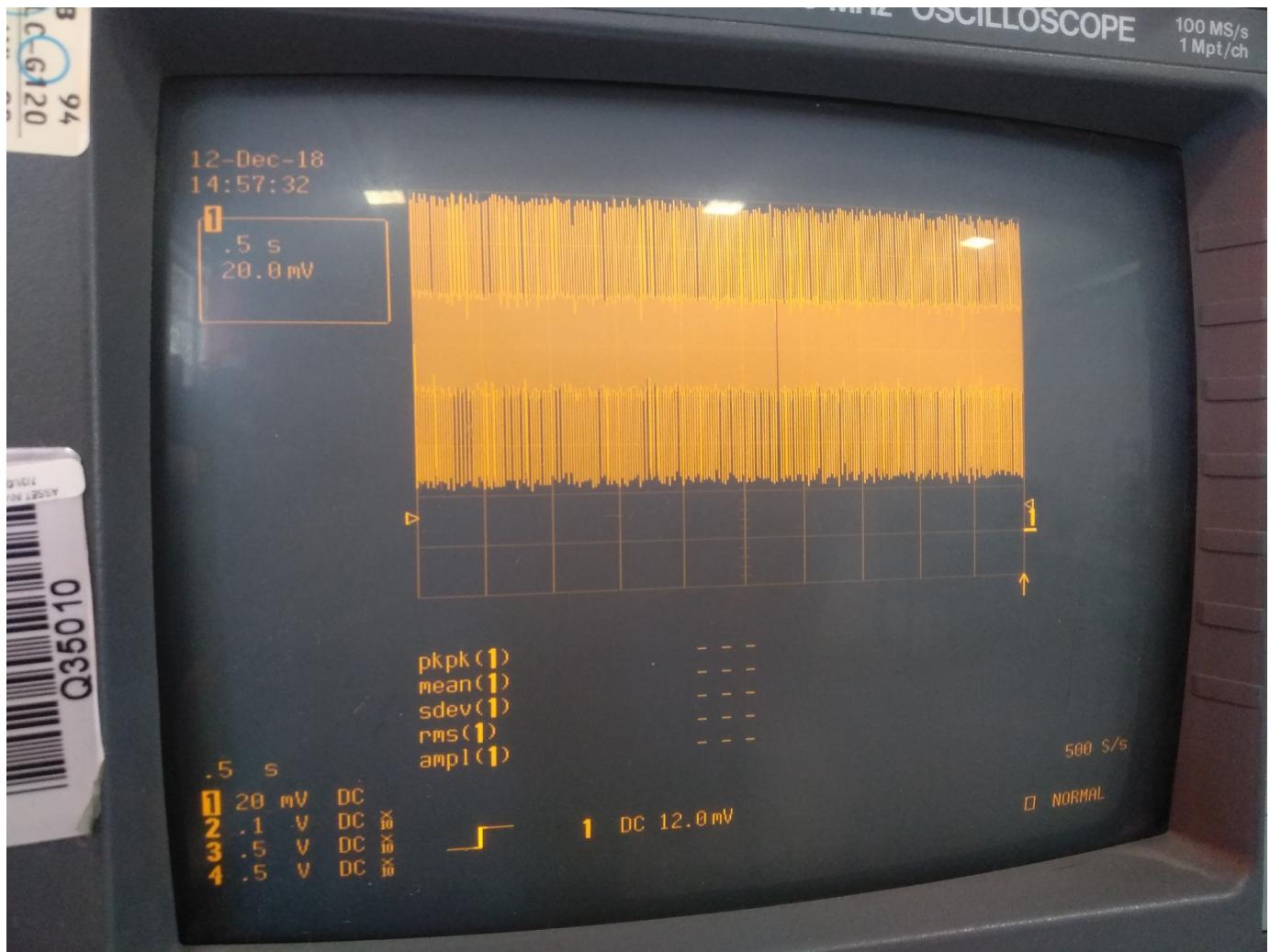
Atmega328p-pu Debugging. How things should be.

1)XTAL without ceramic capacitors

This is how XTAL2 (X2) should look like, without capacitors:

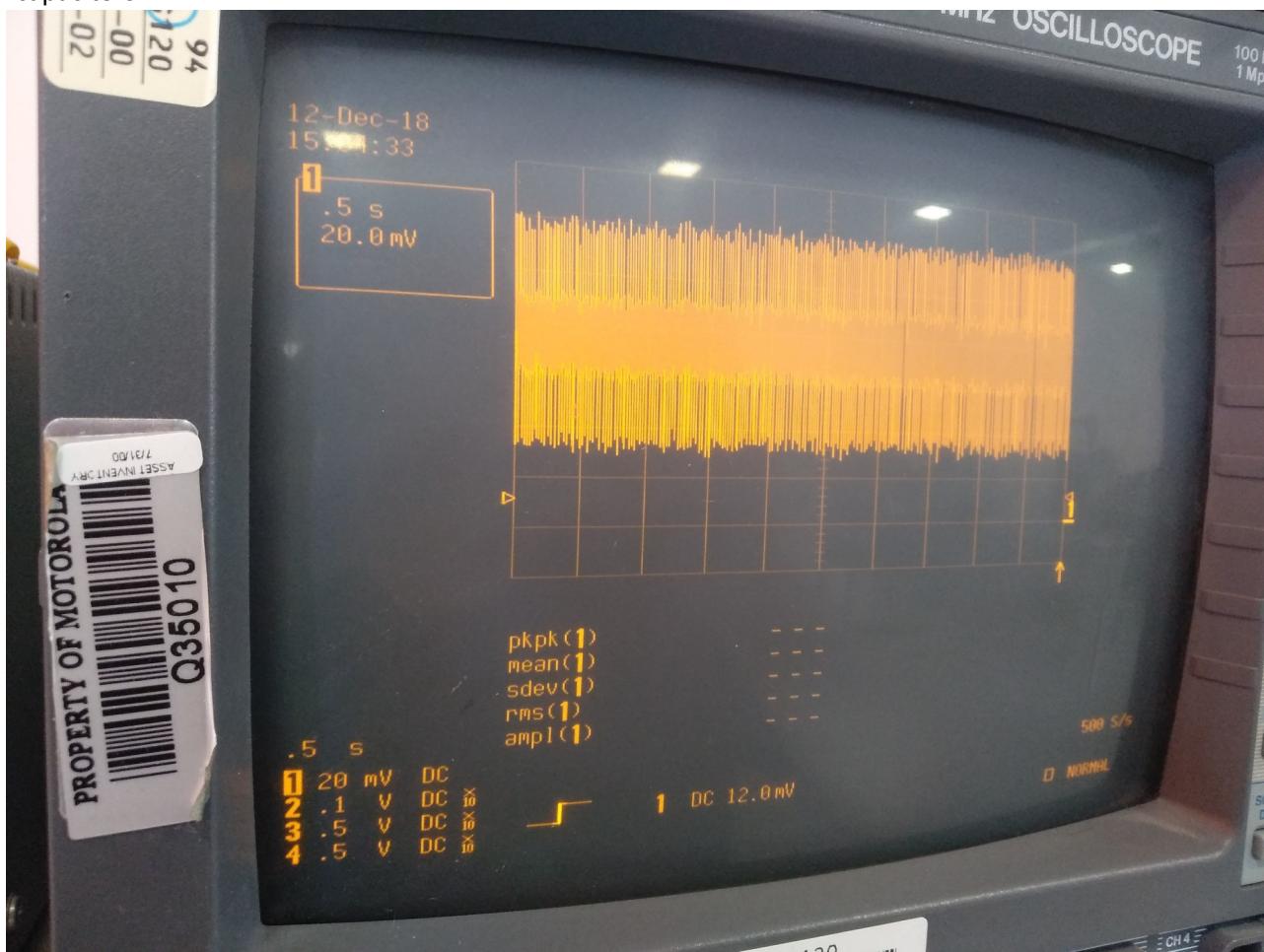


This is how XTAL1 (X1) should look like, without capacitors:

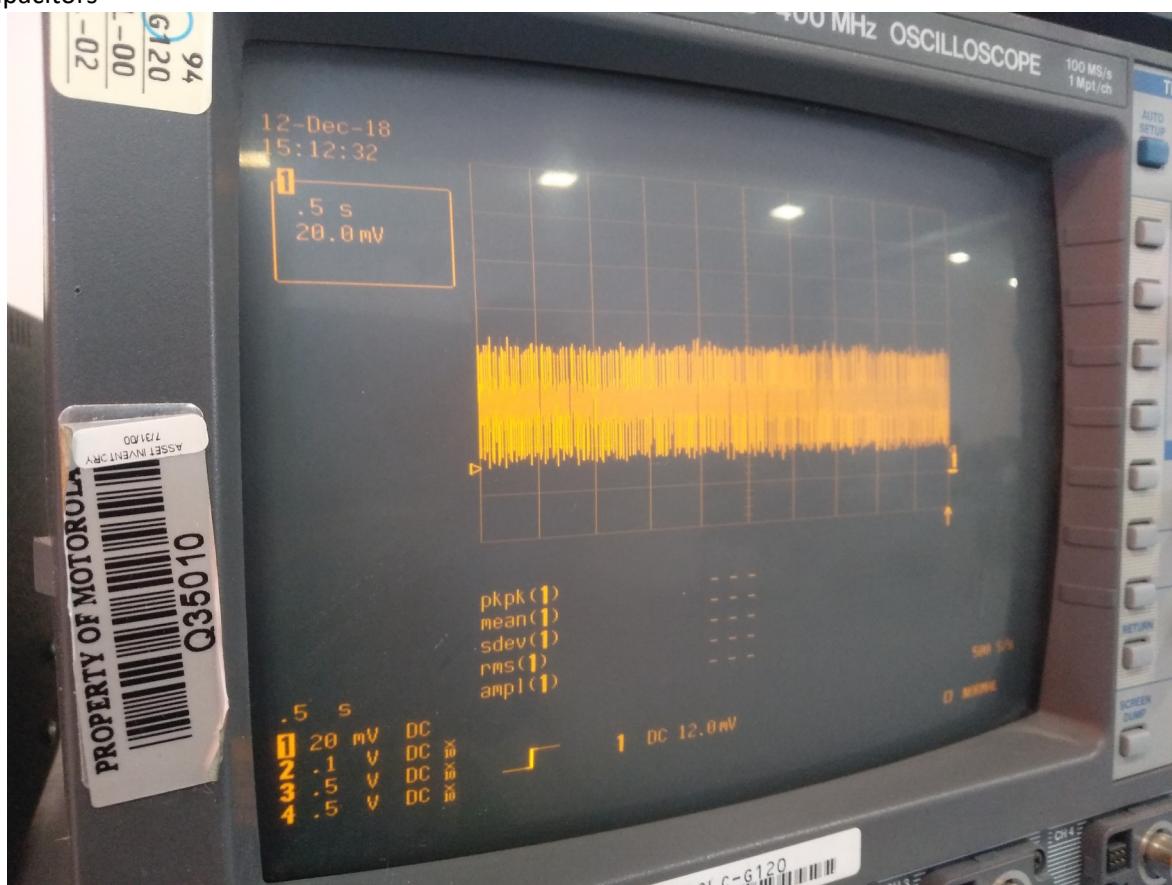


2)XTAL with ceramic capacitors

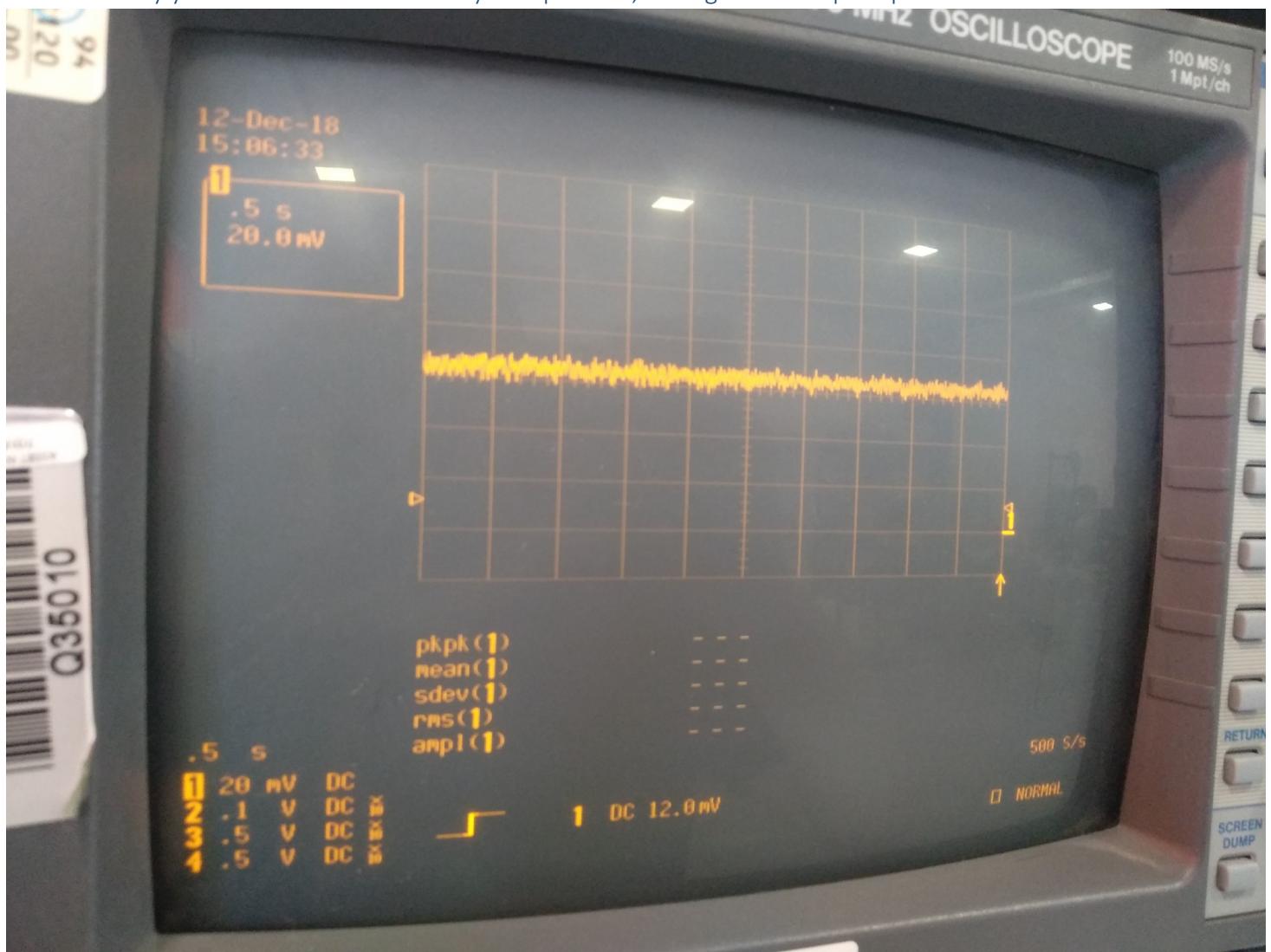
Xtal1 with capacitors



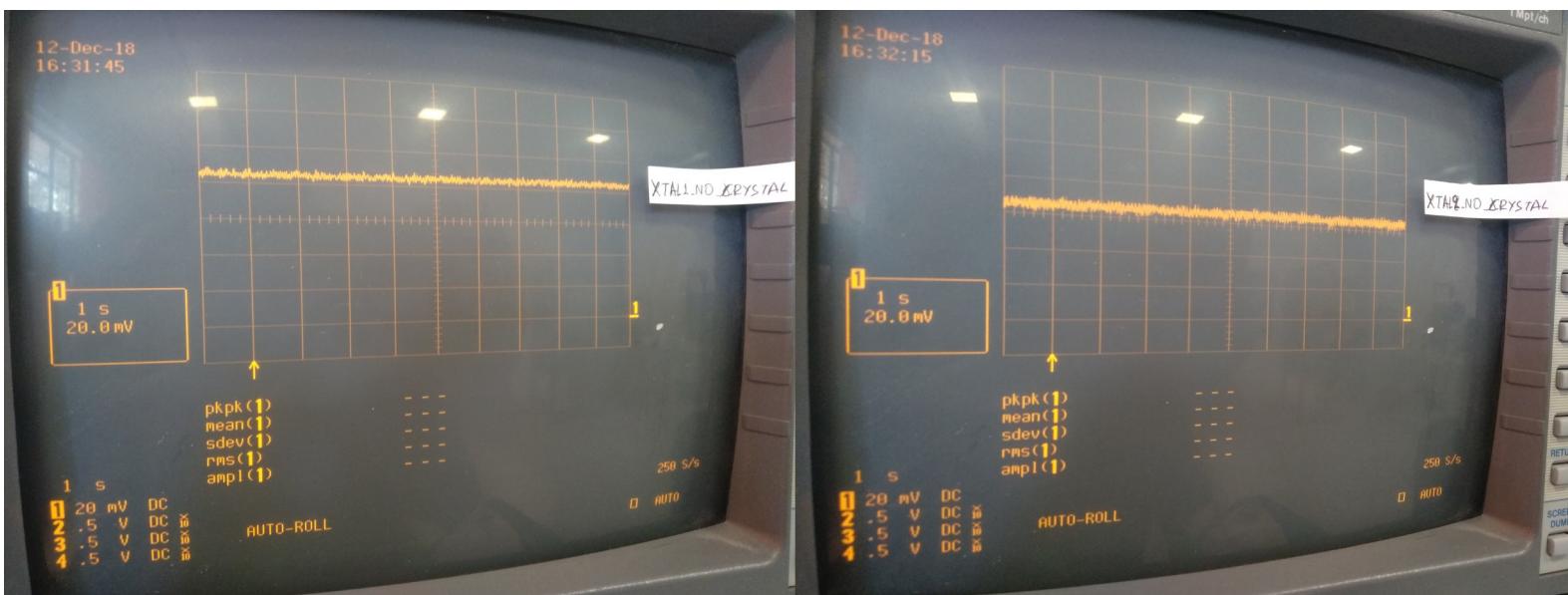
XTAL2 with capacitors



Bonus: Why you should not use electrolytic capacitors, or larger than 22pF capacitors:



But what happens when a crystal is not in place? What does the atmega tries to do? It turns out, it outputs 0.840volts at XTAL1 and 0.6 at XTAL2.

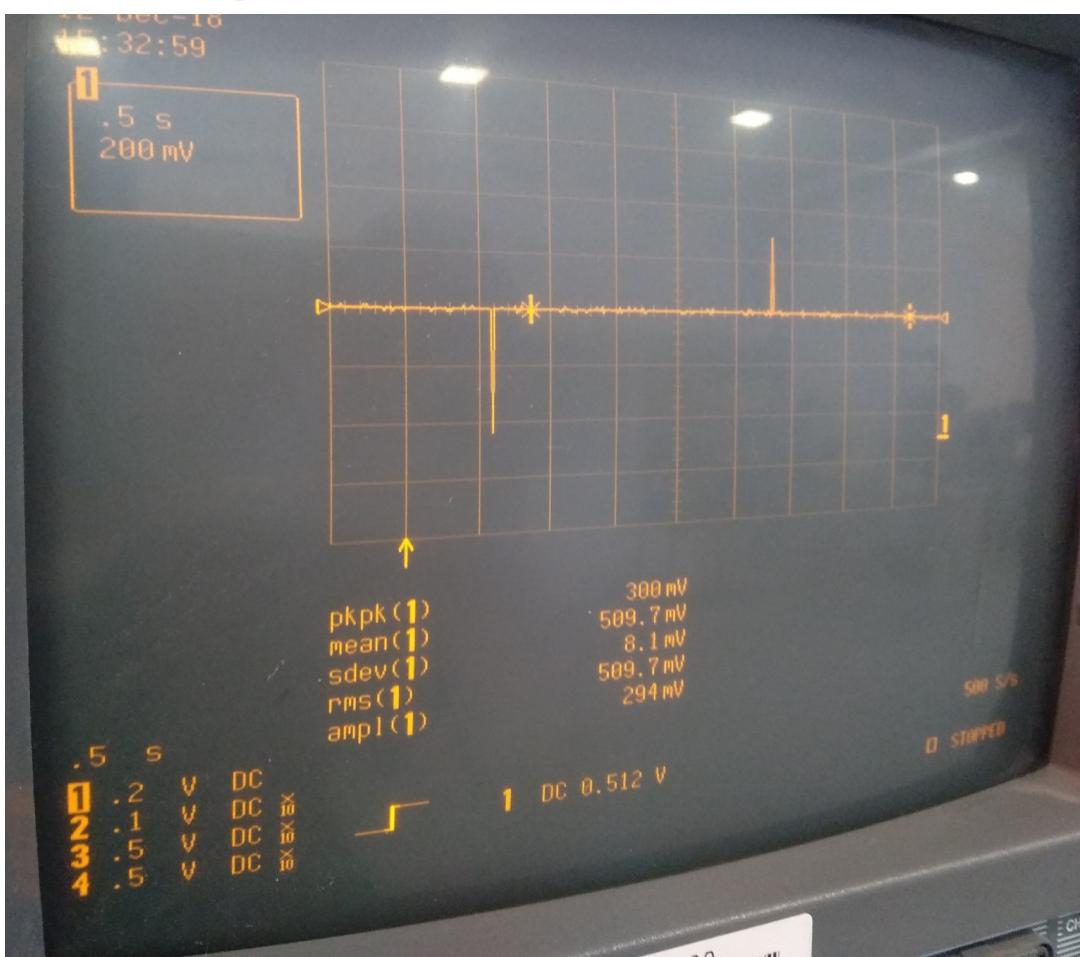
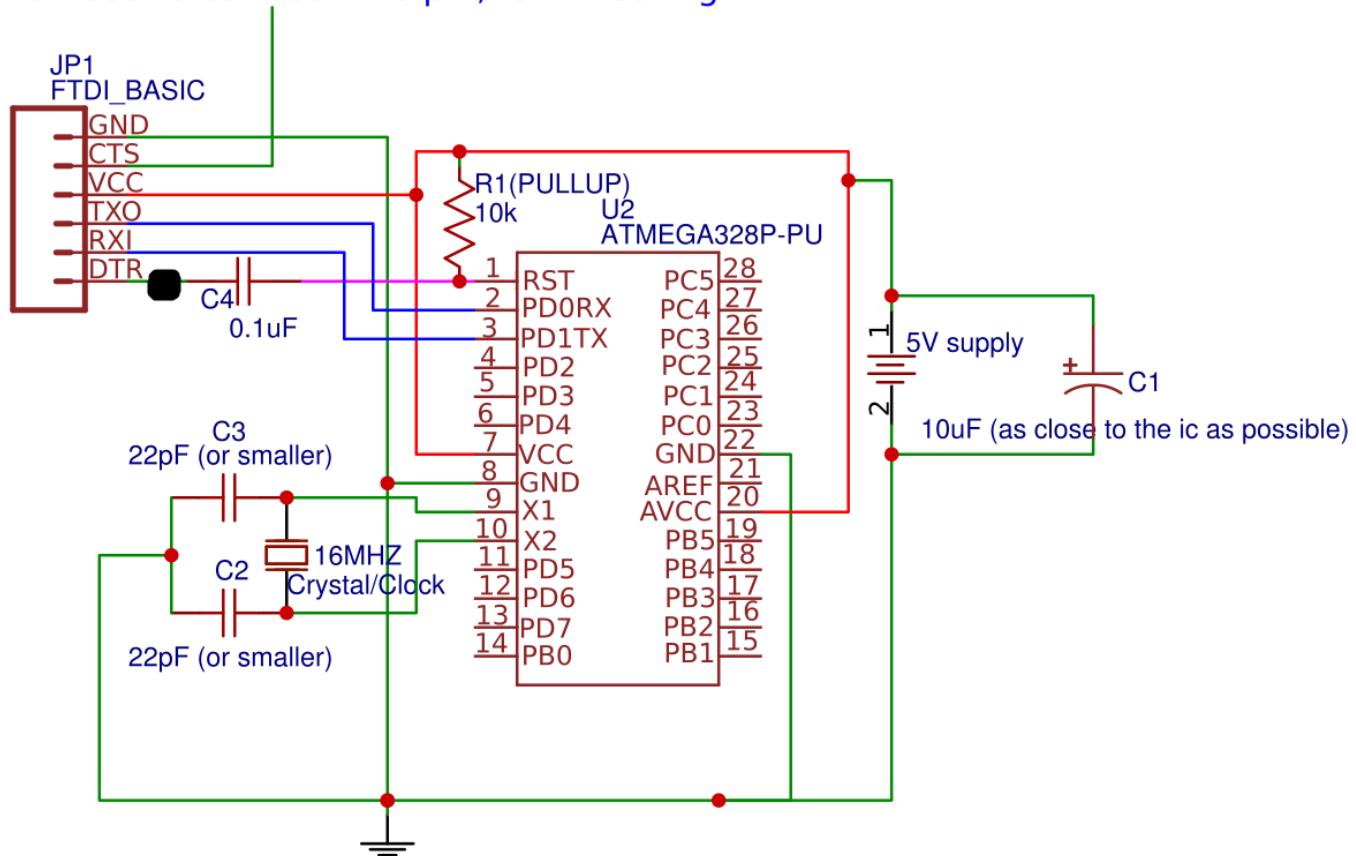


Programming the mcu, oscilloscope on the UART pins

We reconnect the crystal again, (and for the rest of the tutorial the crystal will be connected). And first we check:
 Signals from DTR, before C4: (black dot is where the oscilloscope is connected):

How the signal changes after the C4:(black dot is where the oscilloscope is connected):

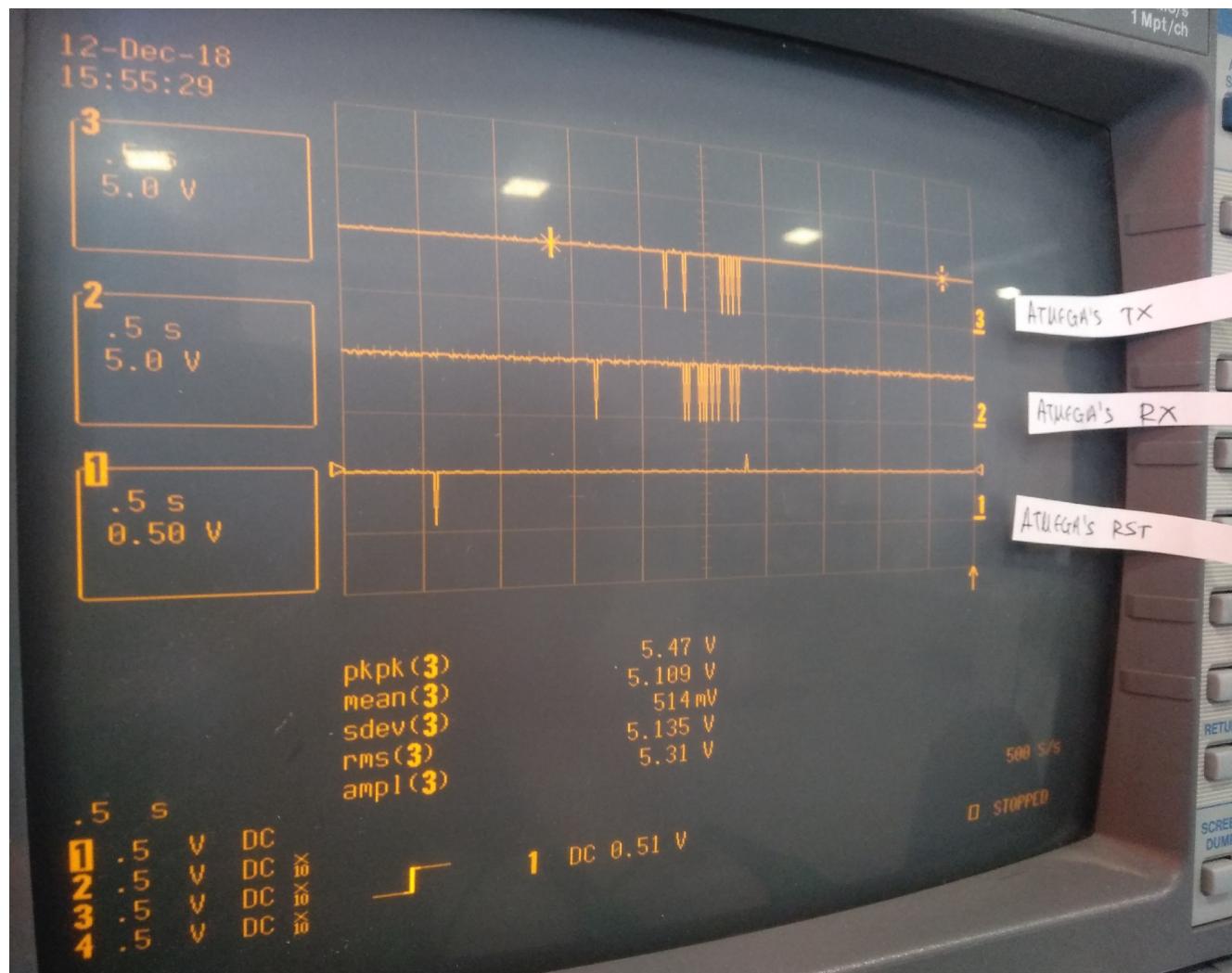
No need to connect this pin, let it floating



This is what the atmega sees.

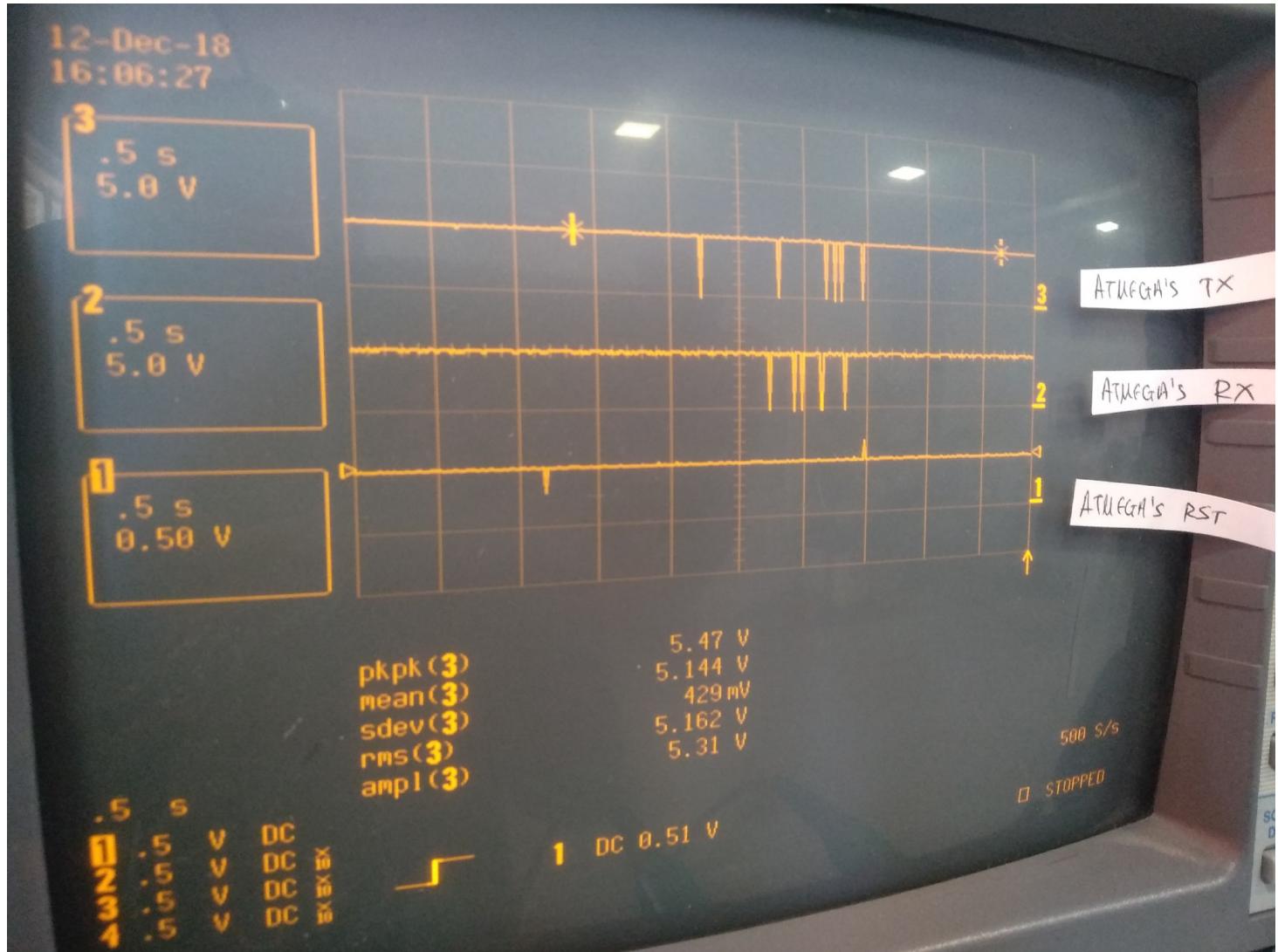
Next, using the arduino ide, the atmega328p-pu will be programmed. With an empty code:

```
void setup() {  
  // put your setup code here, to run once:  
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
}
```



Using atmel studio, to upload an empty program:

```
int main(void)
{
}
```



Tutorial created: 05/Aug/2018

last edit: 15/Nov/2019

By: Christianidis Vasileios

Contact info: christianidis@disroot.org