Table of Contents

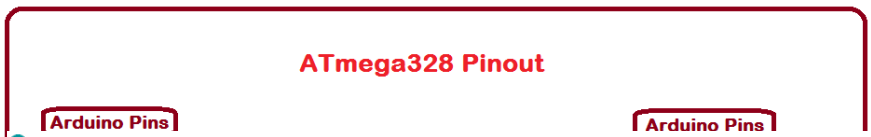
The purpose of this tutorial is to help you use your Atmega328P as a standalone, without the need of an Arduino and with the use of the less needed components to boot it up and run. This will also help you save money on buying a while Arduino board, it will help you reduce the power consumption of a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

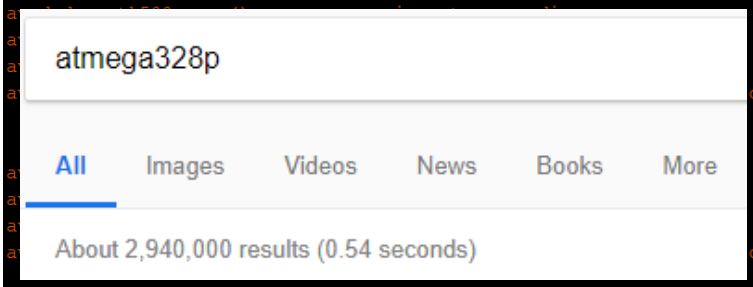
In case you didn’t know, Arduino is called just the platform that hosts the microcontroller, (the Atmega328P is the microcontroller).



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

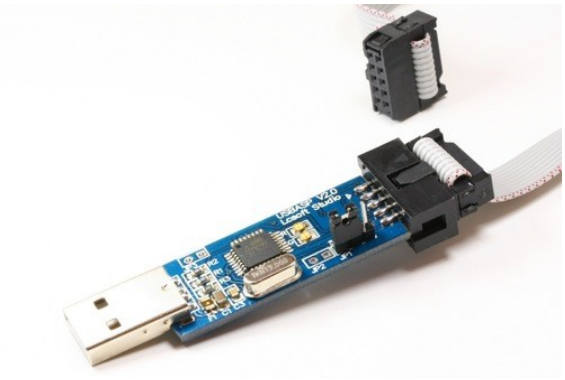
2)

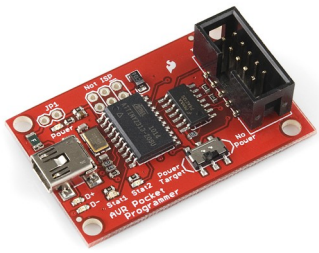


https://www.arduino.cc/en/Hacking/Bootloader

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).

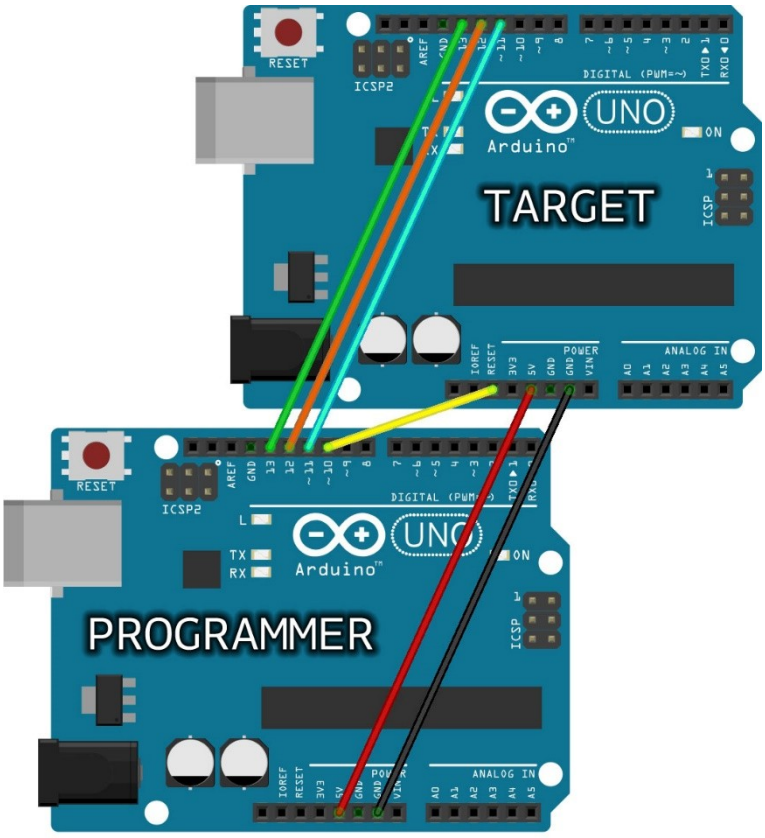
I will just show you how an **AVR programmer** looks like.





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.

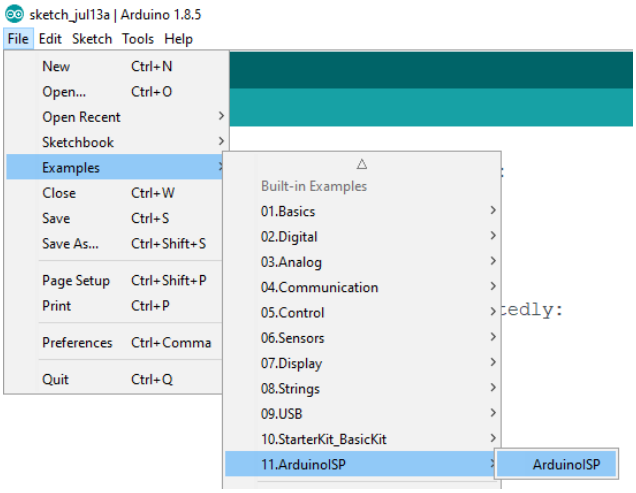
So, one way to burn a bootloader to your atmega, is to put your atmega that doesn't have a bootloader on an Arduino board, and using a second Arduino board (with a second atmega that has a bootloader on it and works just fine) as the programmer! So the second Arduino board that works just fine will be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



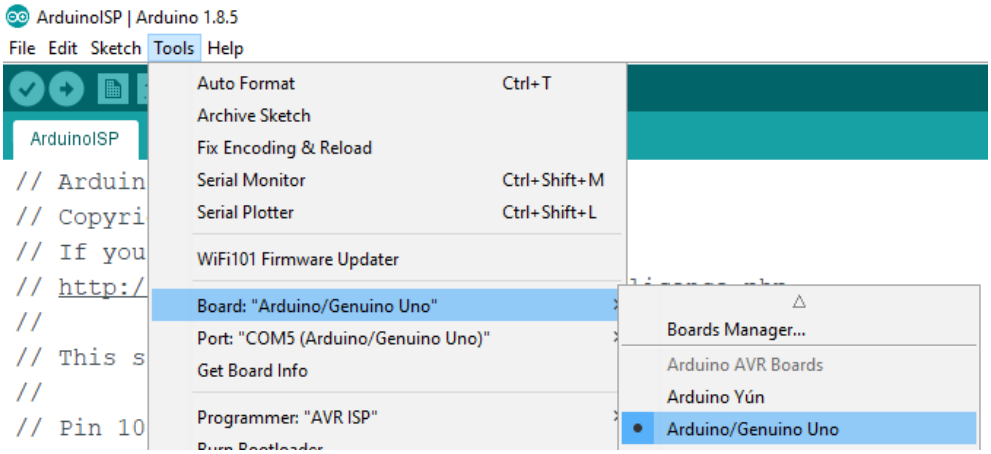
On the **target** we will put the atmega without bootloader and on the **programmer** we will put our atmega that works correctly.

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:

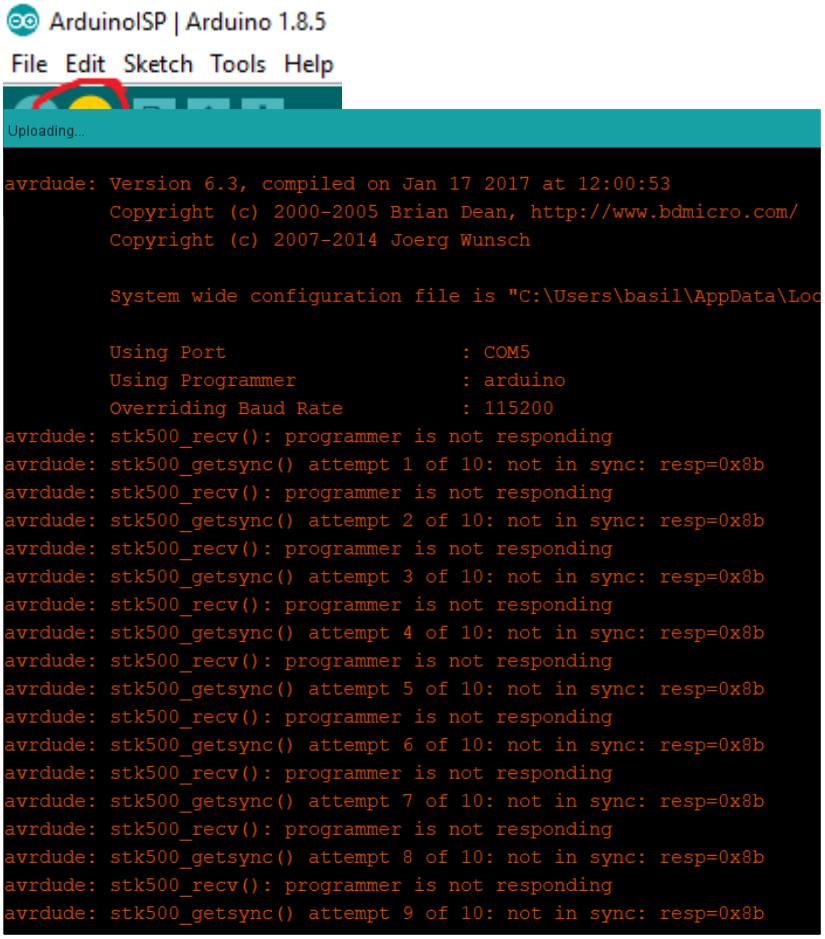


Don’t forget to select the right board:



And the right COM port:

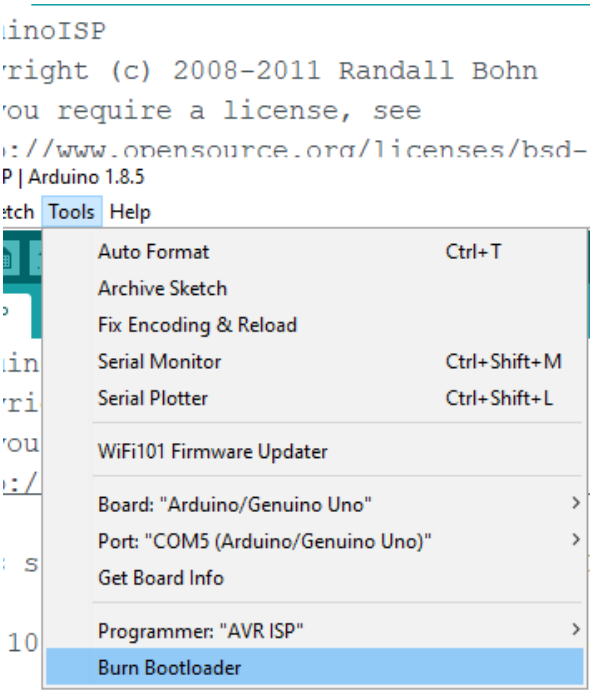
Click upload



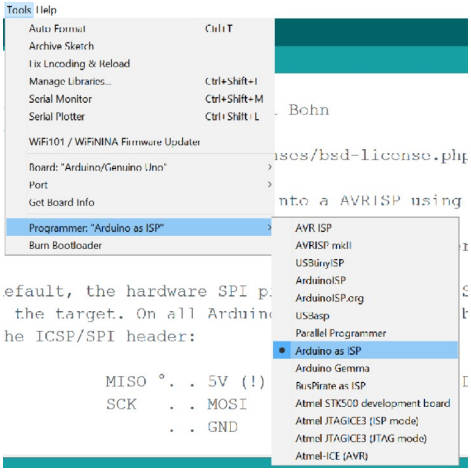
Obviously, if you accidentially tried to upload the sketch to the wrong board (aka to the Arduino board without bootloader) you will get the error So if you did it correctly, it should have uploaded like any other program.

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”



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.



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, ju[redacted] 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:

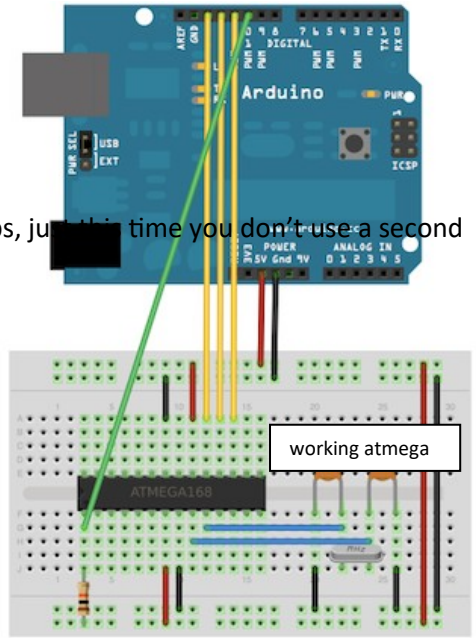
Burning the Bootloader

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:

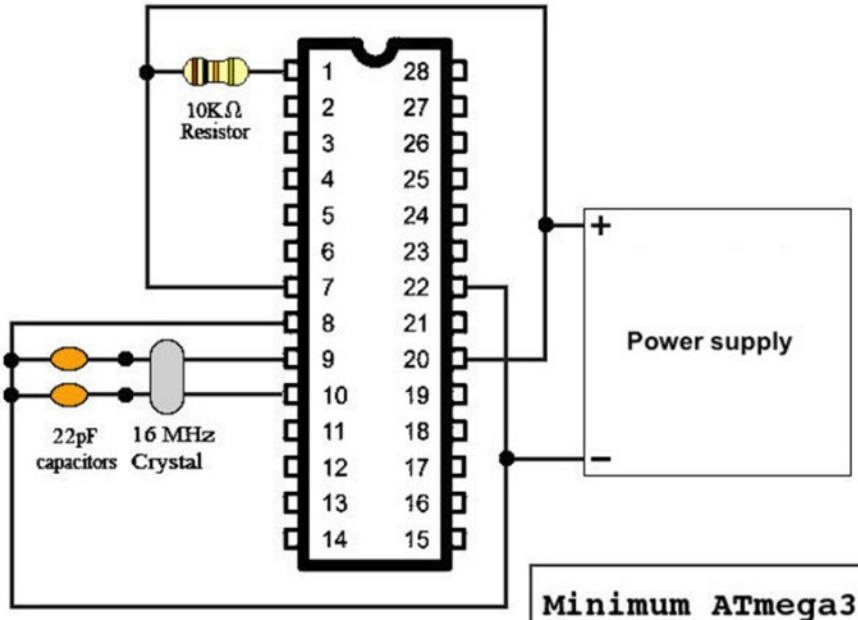
1. Upload the ArduinoISP sketch onto your Arduino board. (You'll need to select the board and serial port from the Tools menu that correspond to your board.)
2. Wire up the Arduino board and microcontroller as shown in the diagram to the right.
3. Select "Arduino Duemilanove or Nano w/ ATmega328" from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

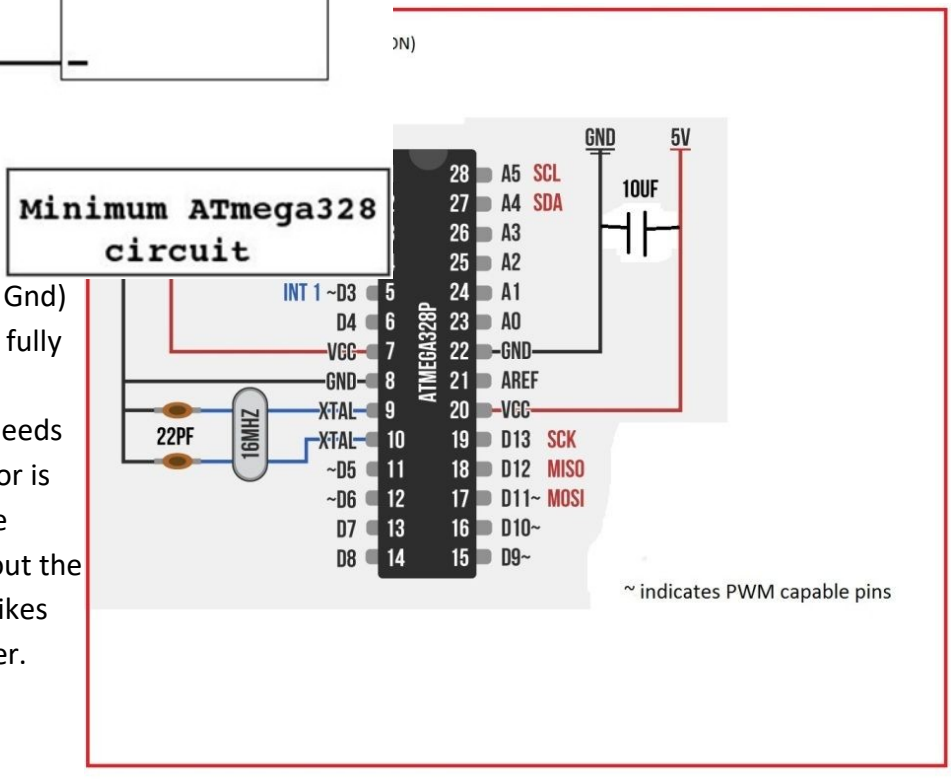


Using an Arduino board to burn the bootloader onto an ATmega on a breadboard.

Atmega328P with bootloader on a breadboard



nake the atmega work properly.



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.

PICTURE NUMBER: 1.2

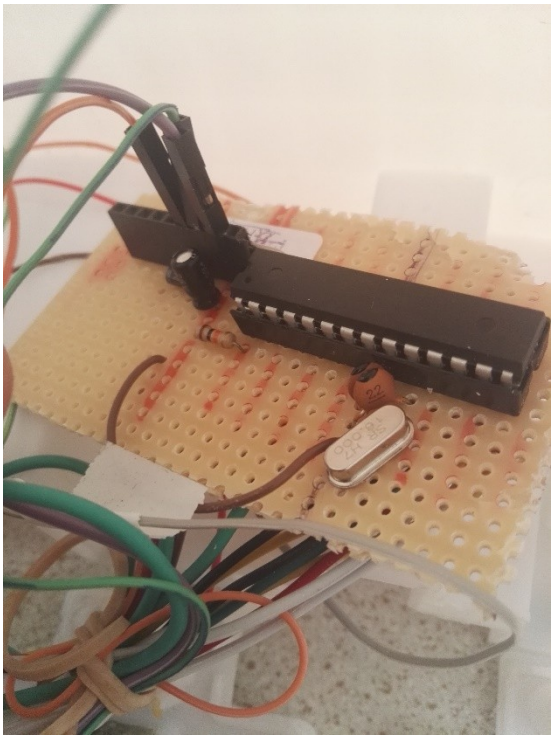
Note that: Atmega works on these voltage ranges:

7.4 ATmega328P

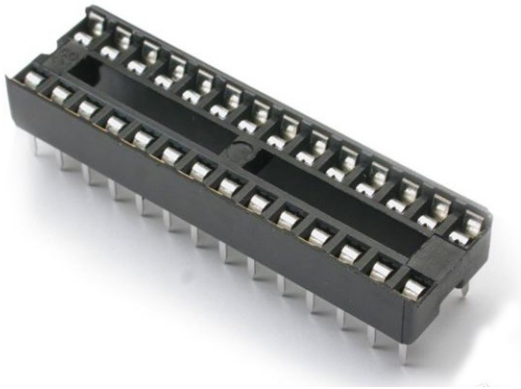
Speed (MHz)[3]	Power Supply	Ordering Code[2]	Package[1]	Operational Range
20	1.8 - 5.5	ATmega328P- AU ATmega328P- MU ATmega328P- PU	32A 32M1-A 28P3	Industrial (-40°C to 85°C)

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

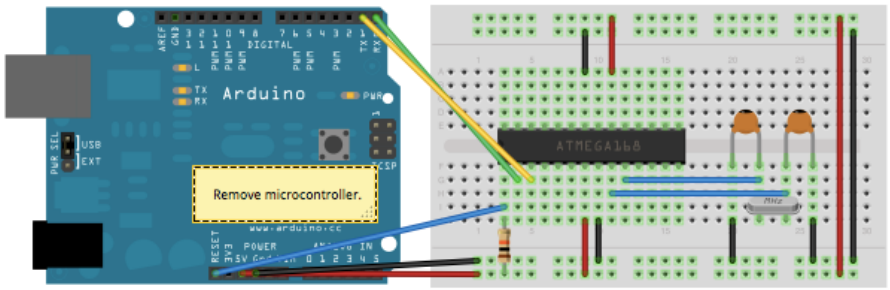


After you check that your breadboard prototype works, you may want to make it more permanent, by soldering all the components on a pcb board like the picture. Take care, do not solder directly on the atmega, or you will risk of burning it. instead, do what Arduino board does. Without using the atmega, 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).

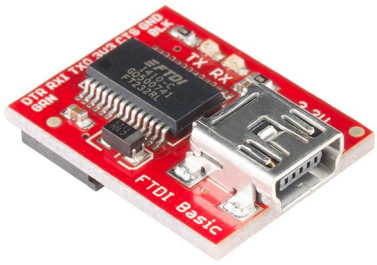


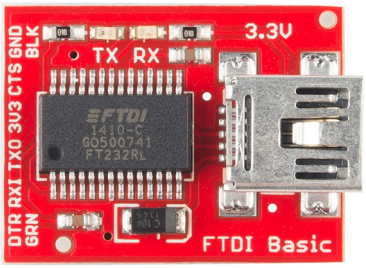
Uploading sketches to an ATmega on a breadboard. Remember to remove the microcontroller from the Arduino board!

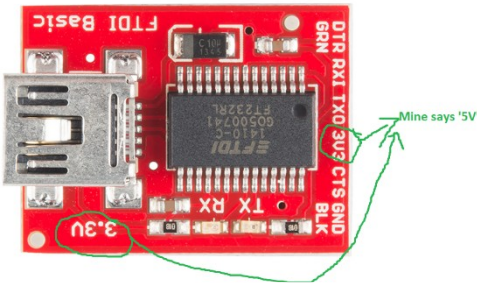
3rd way: Using an FTDI breakout.

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

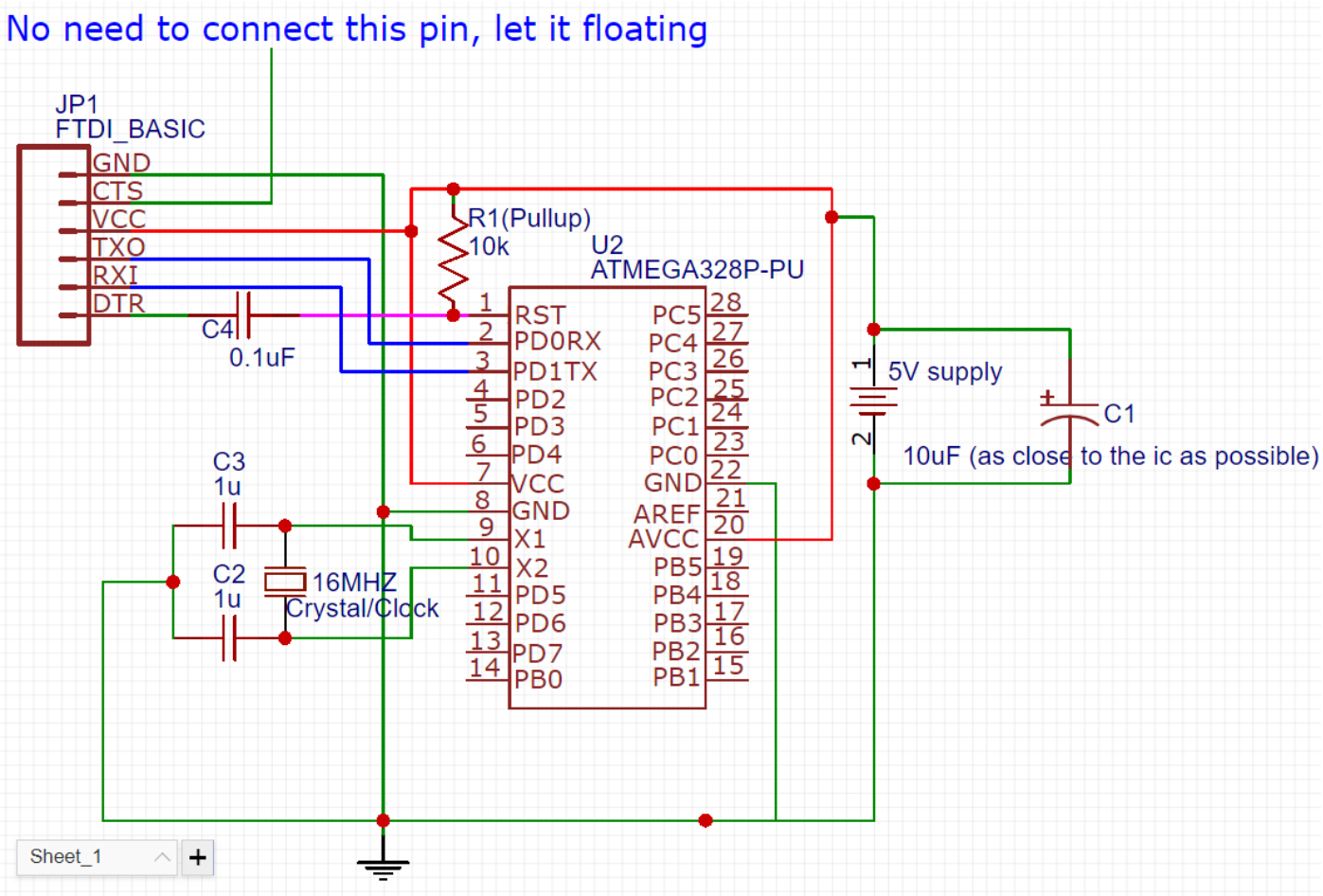






Now, here is a little difference I found out in my ftdi.
mine says '5V instead of 3v'. Make sure to use the one that says 5V, you will need it to power the Arduino.

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:

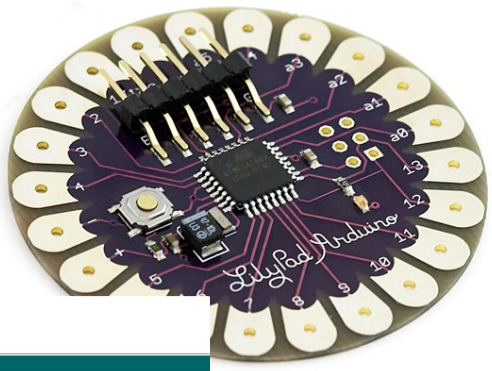


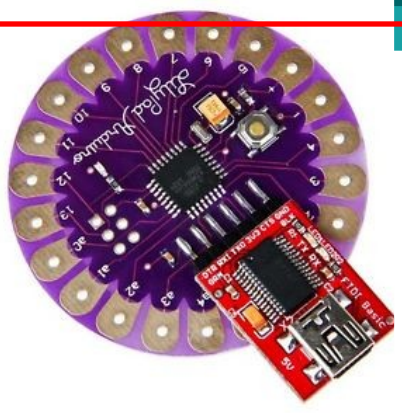
The 0.1uF capacitor is crucial, so that atmega wont reset itself repeatedly.

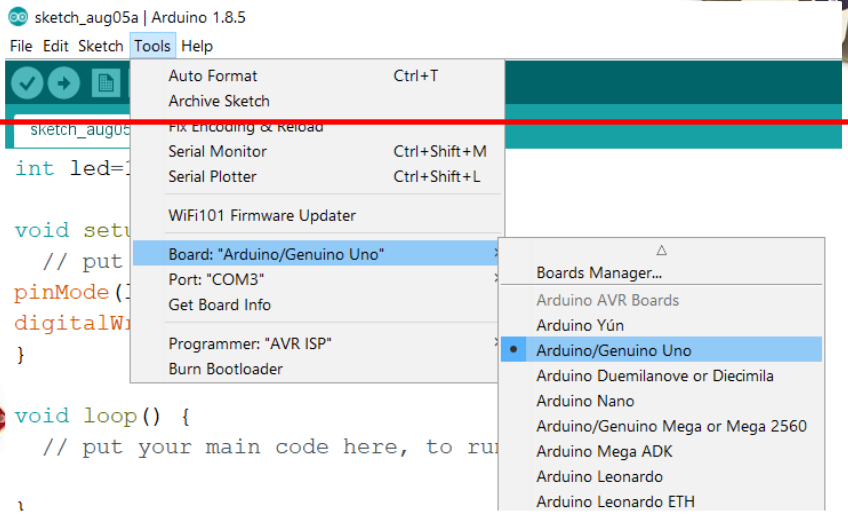
And, of course because the real-life FTDI_BASIC also has a usb port to connect it to the computer, the actual circuit looks like this:

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.



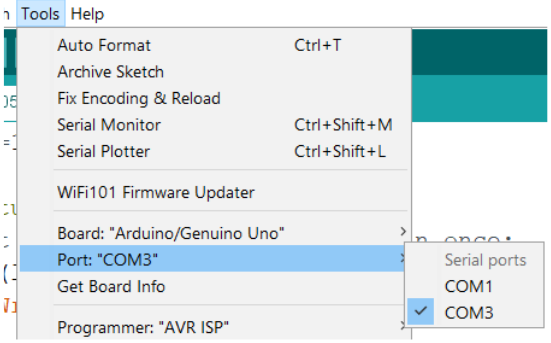




Once you have your FTDI connected on the atmega and the FTDI connected to the computer through usb,

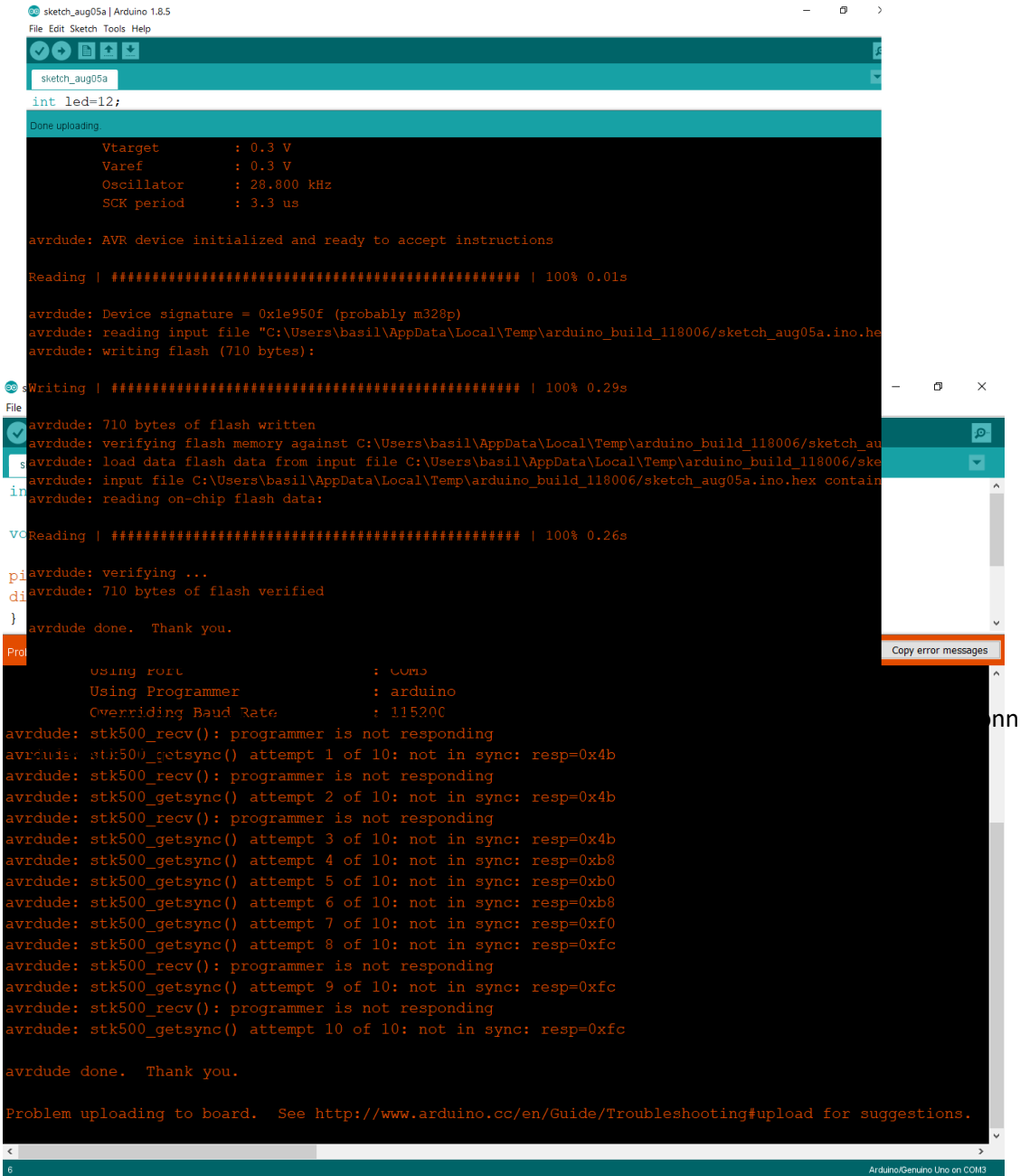
you can open the Arduino IDE, choose the board for the Arduino Uno. The ide will think that there is an arduino uno connected on it, and it will send all the code to it (to the FTDI).

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:



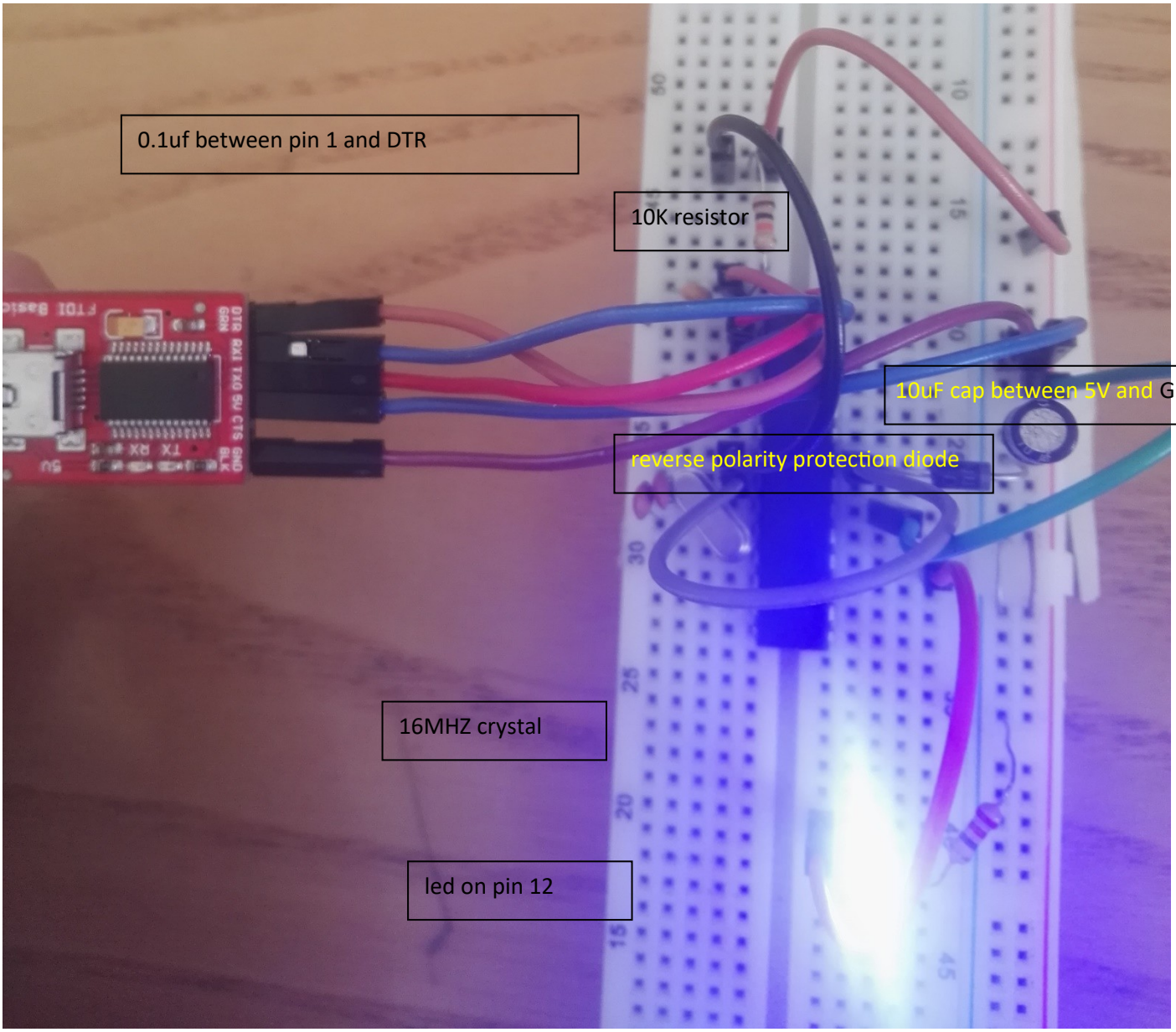
nnect the timer (crystal), it will

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:

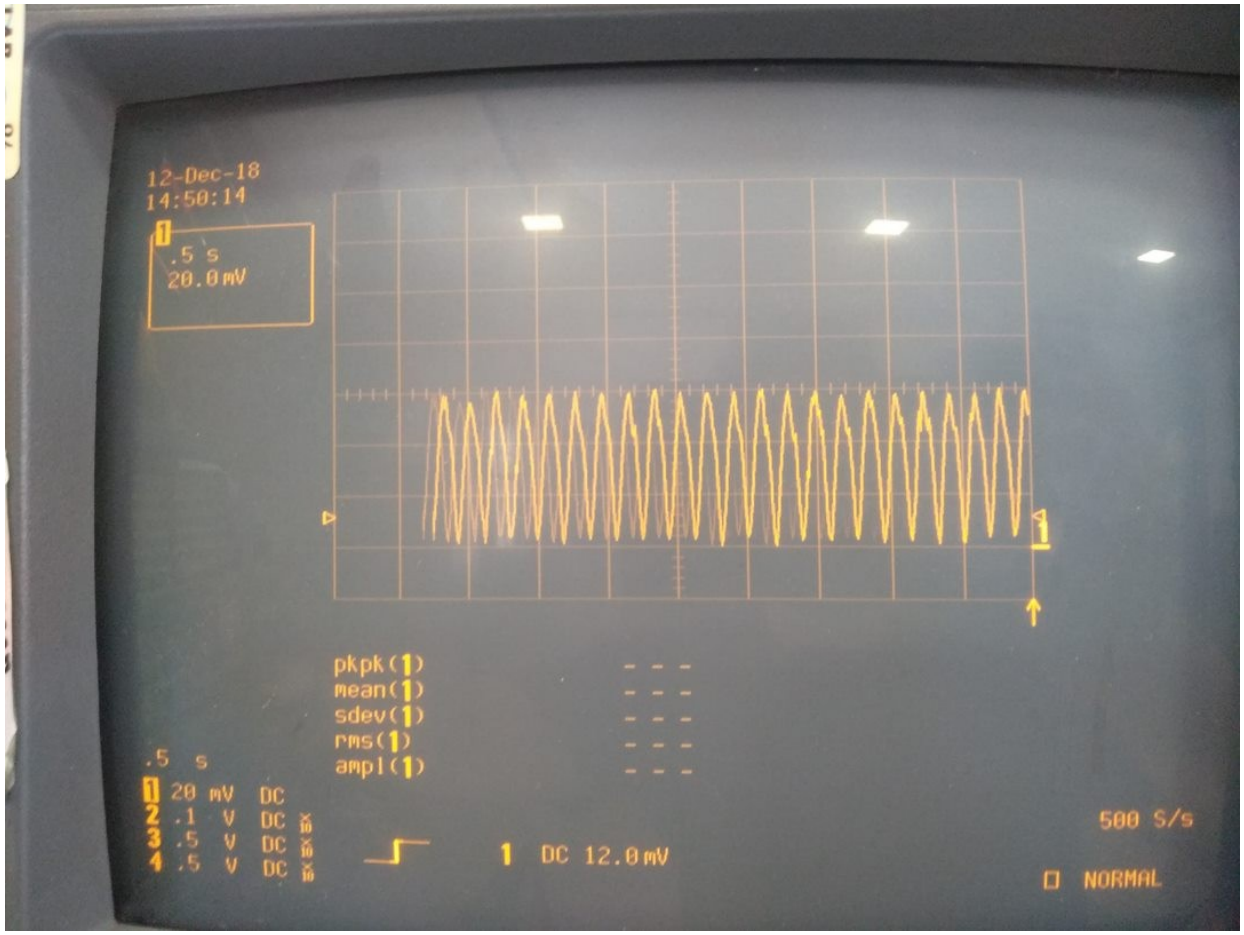
```
int led=12;
void setup() {
  // put your setup code here, to run once:
pinMode(led,OUTPUT);
digitalWrite(led,1);
}
void loop() {
  // put your main code here, to run repeatedly:
}
```

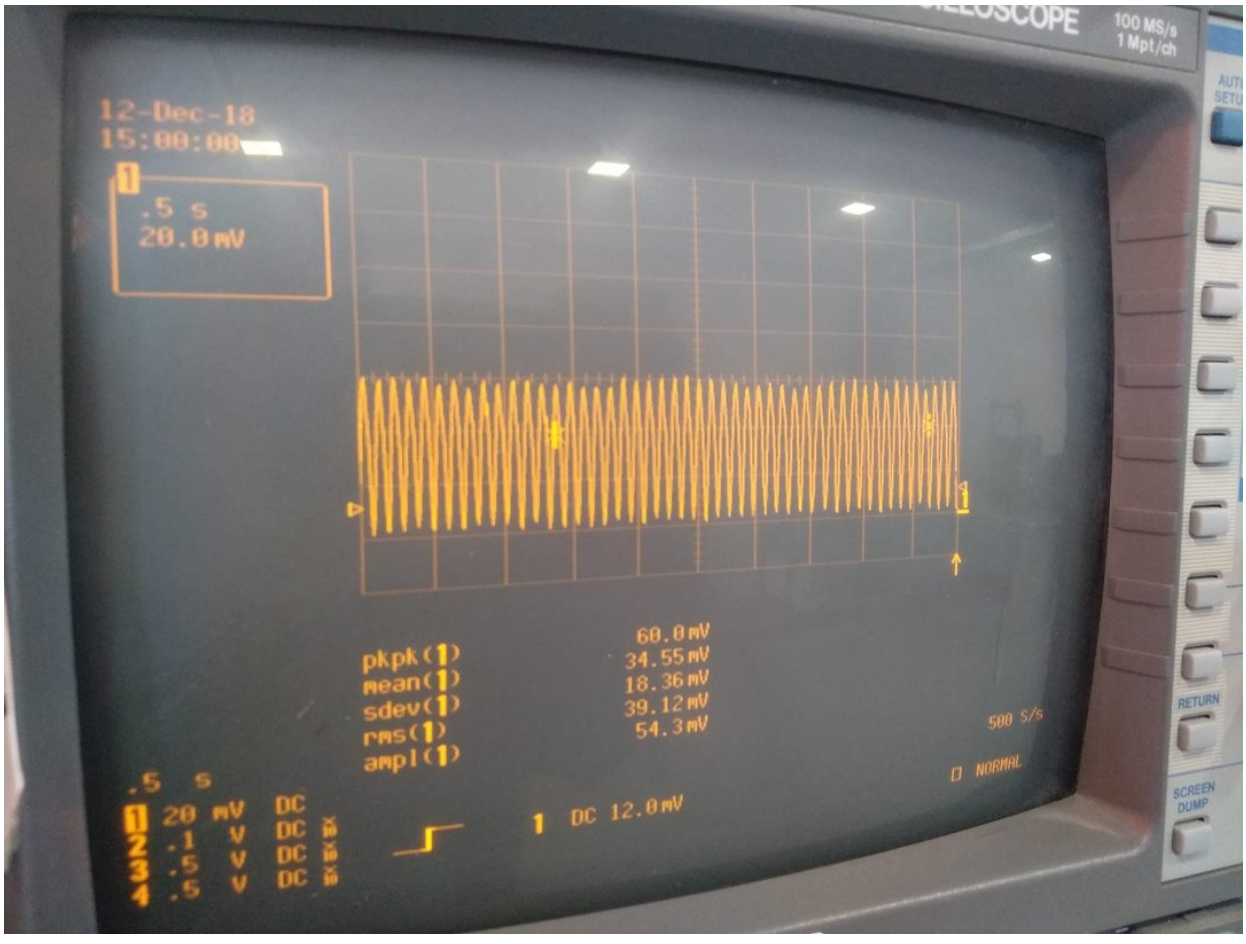
How mine looks like in the end:


0.1uf between pin 1 and DTR
10K resistor
10uF cap between 5V and Gnd of FTDI
reverse polarity protection diode
16MHZ crystal
led on pin 12

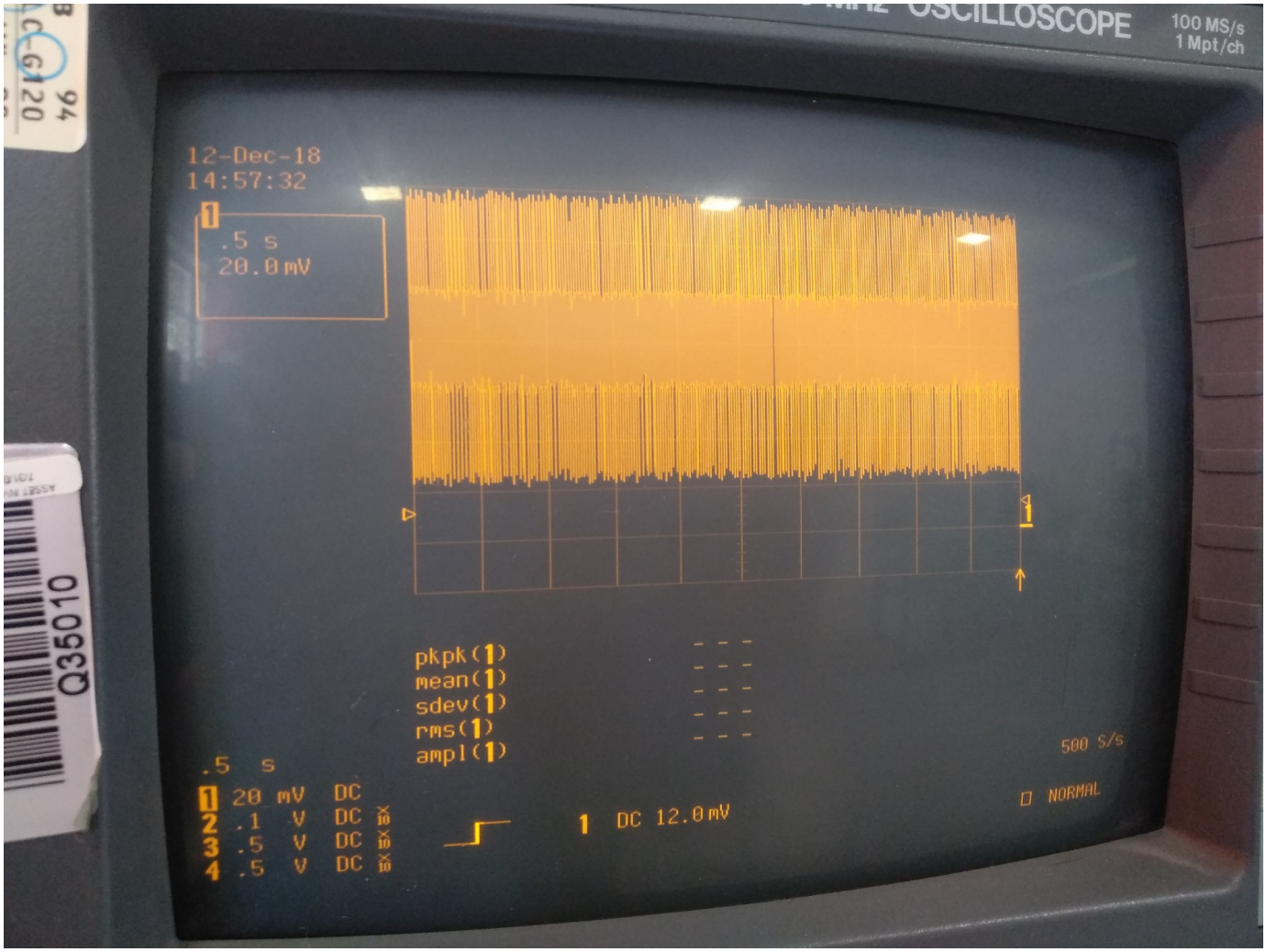
Atmega328p-pu Debugging. How things should be.
1)XTAL without ceramic capacitors

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



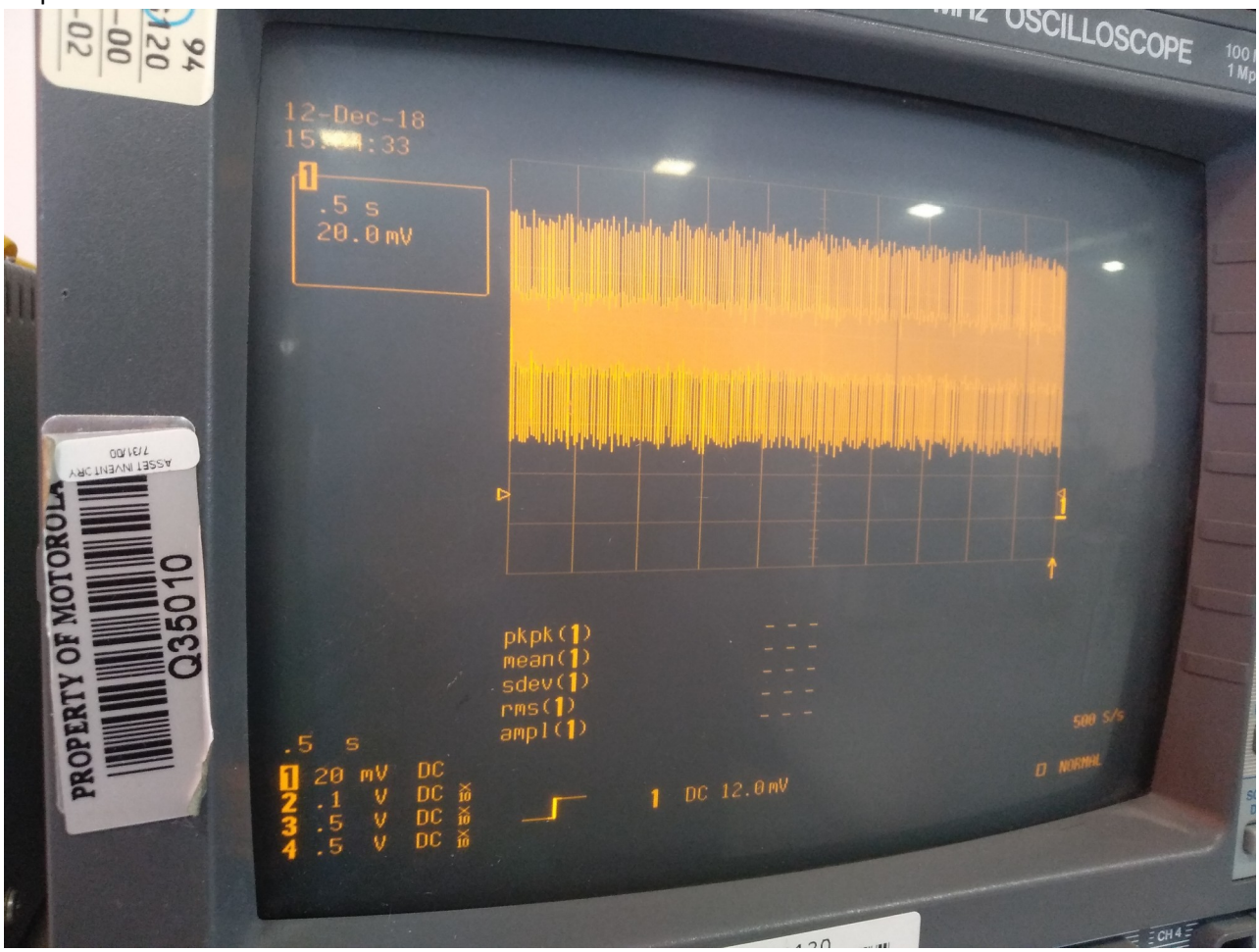


This is how XTAL1 (X1) should look like, without capacitors:

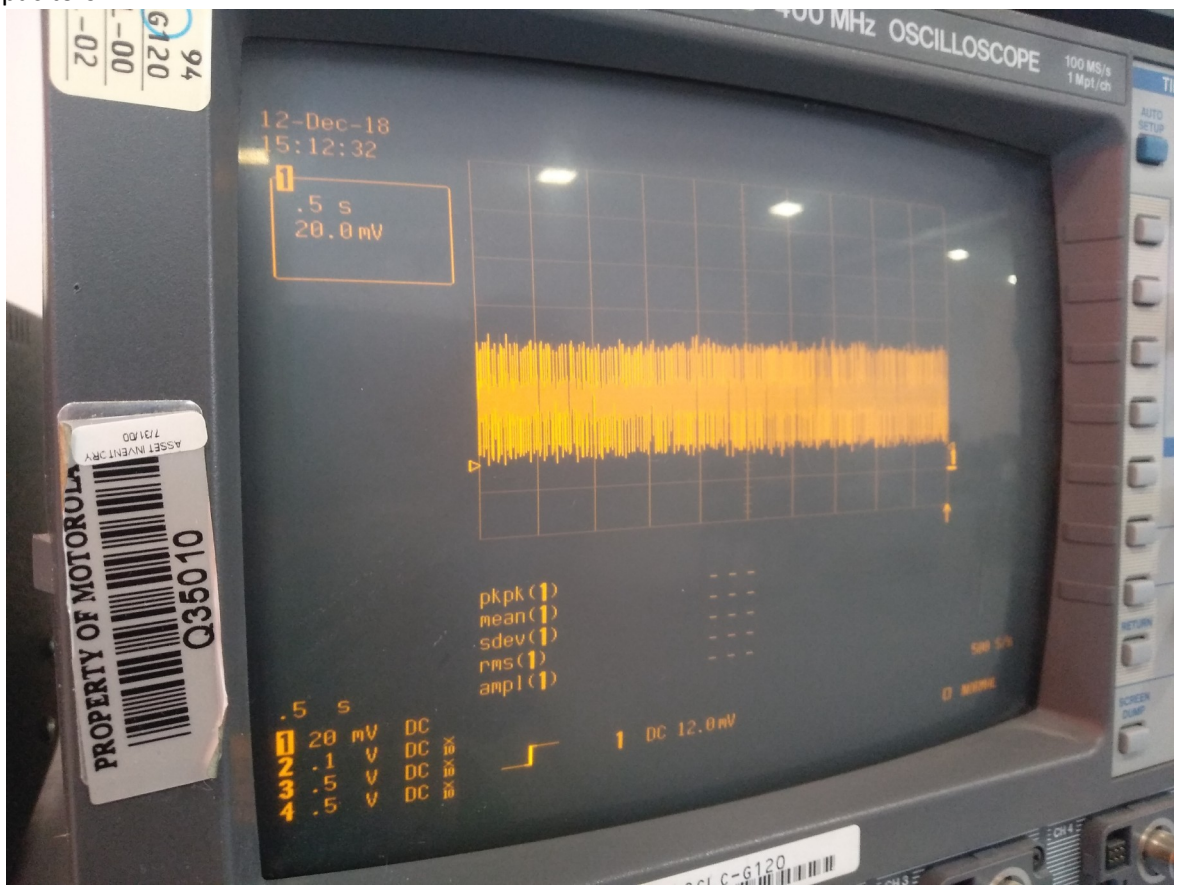


2)XTAL with ceramic capacitors

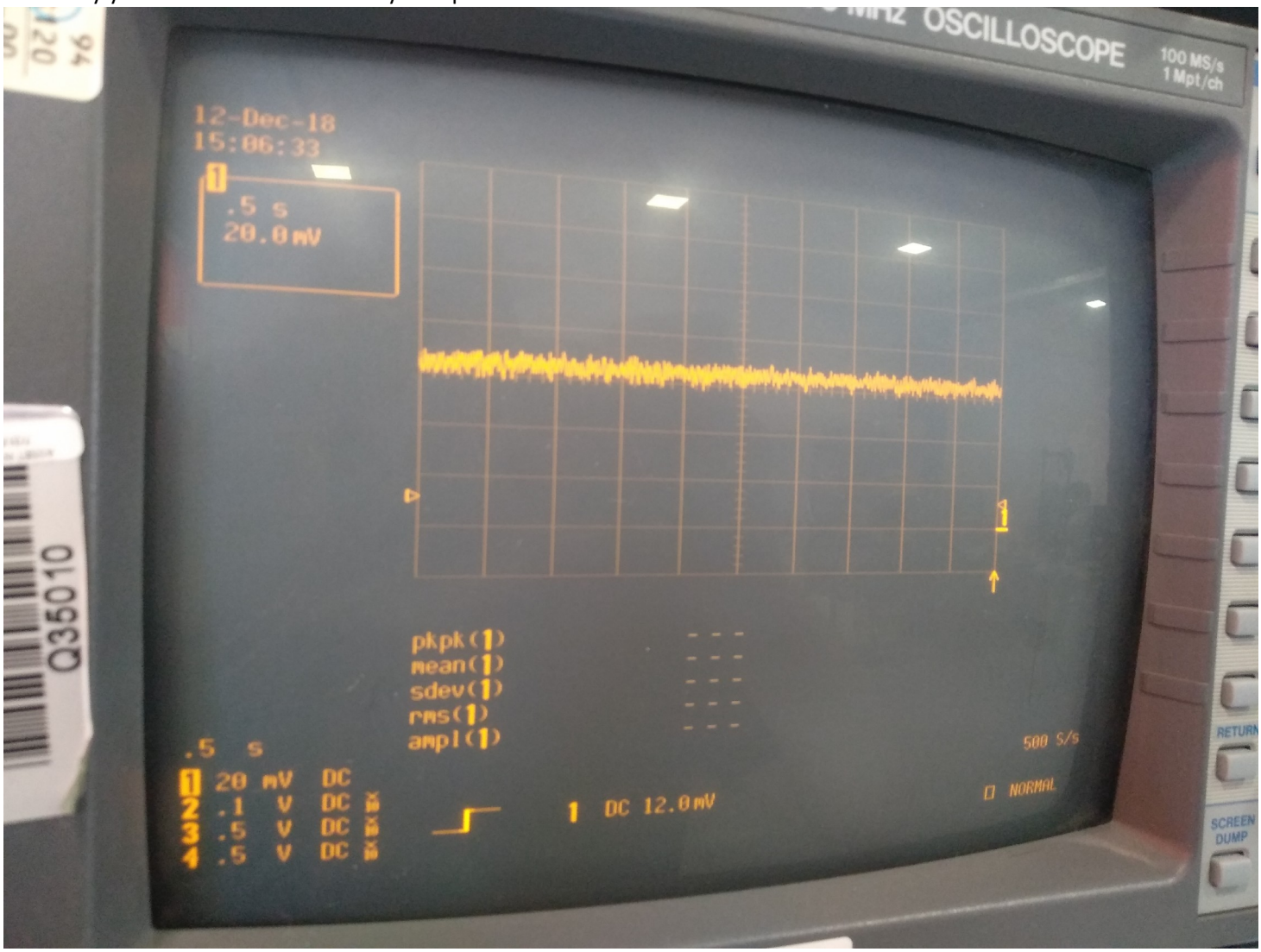
Xtal1 with capacitors



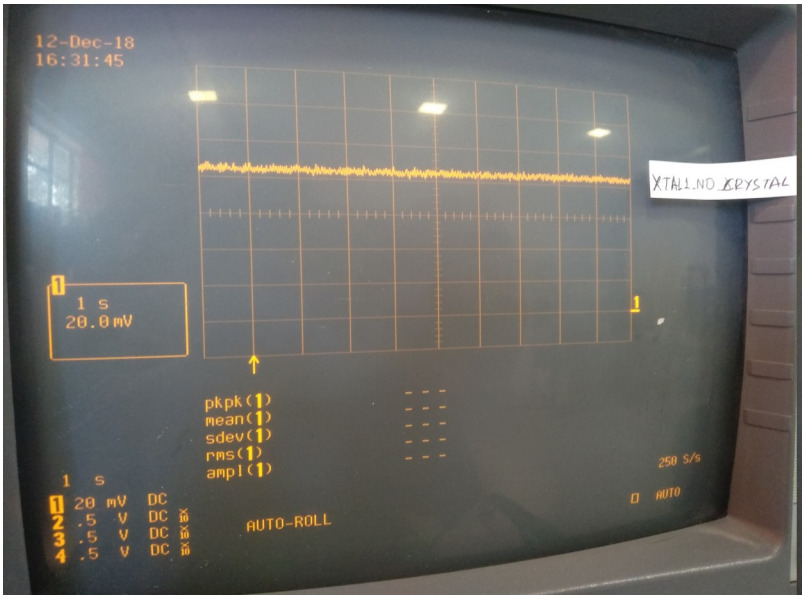
XTAL2 with capacitors

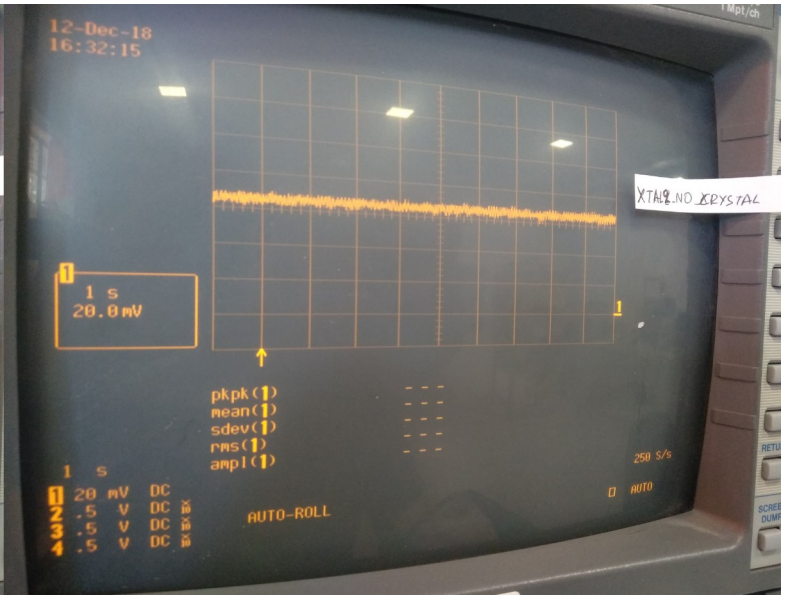


Bonus: Why you should not use electrolytic capacitors:

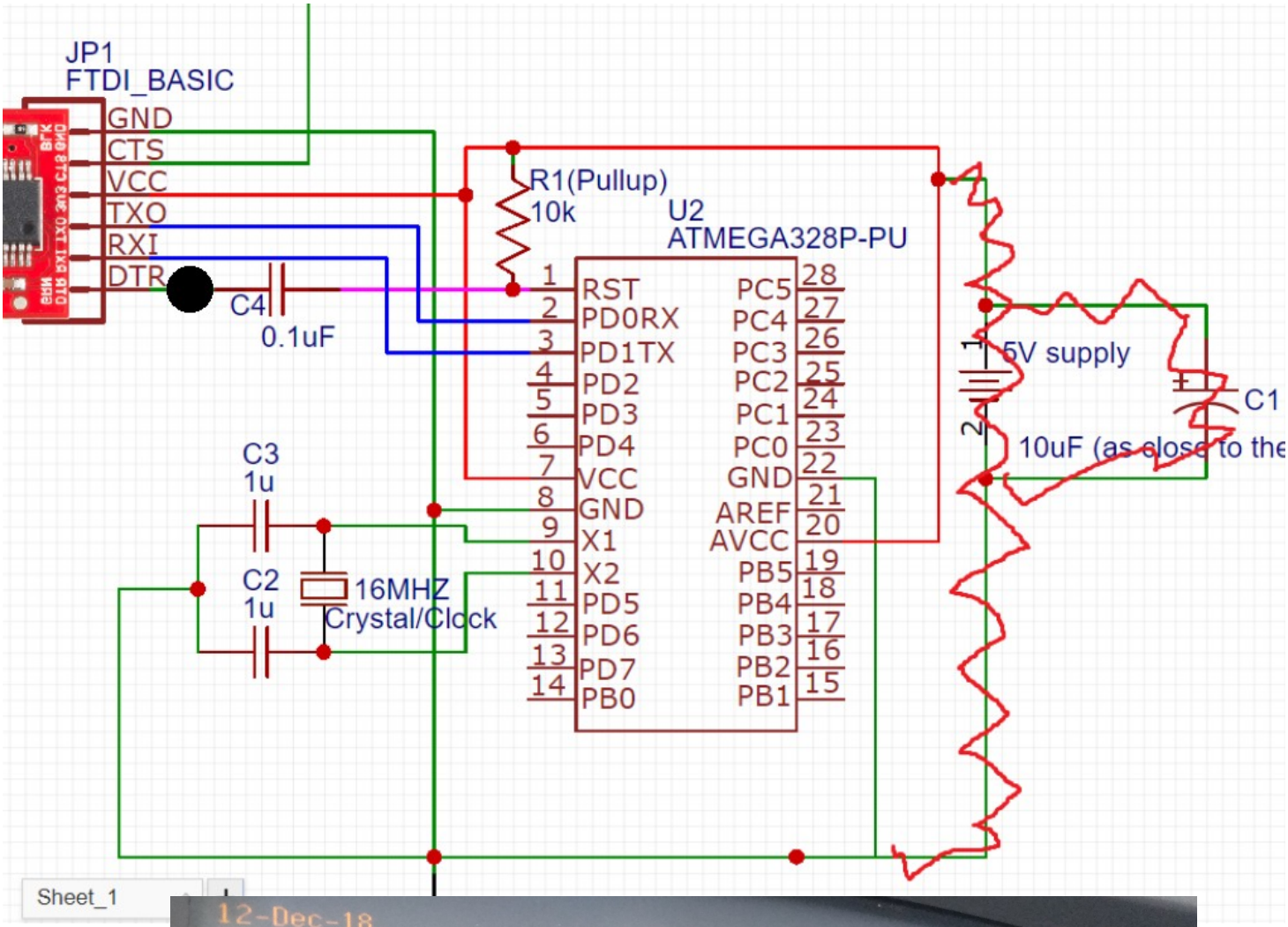


But what happens when a crystal is not in place? What does the atmega tries to do? It turns out, it outputs 0.840volts at XTAL1 and 0.6 at XTAL2.

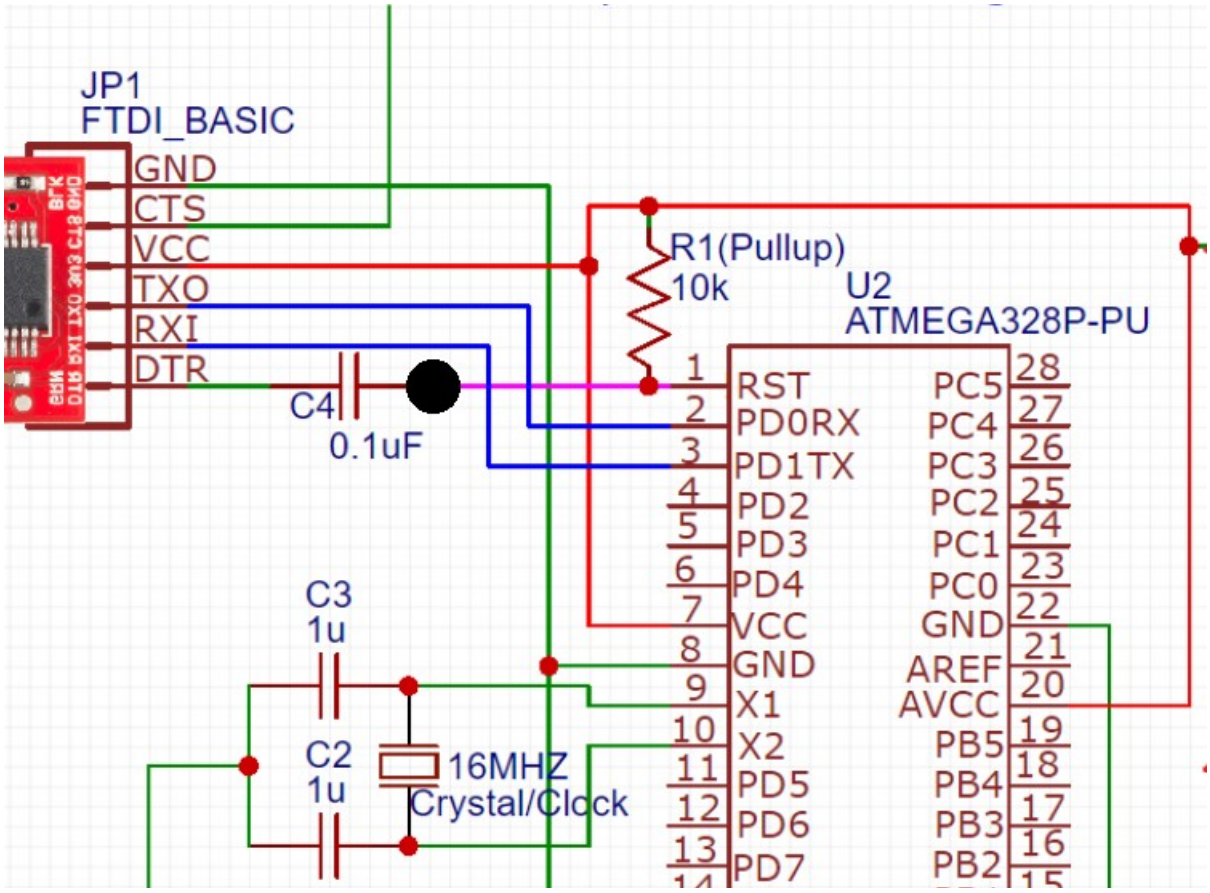


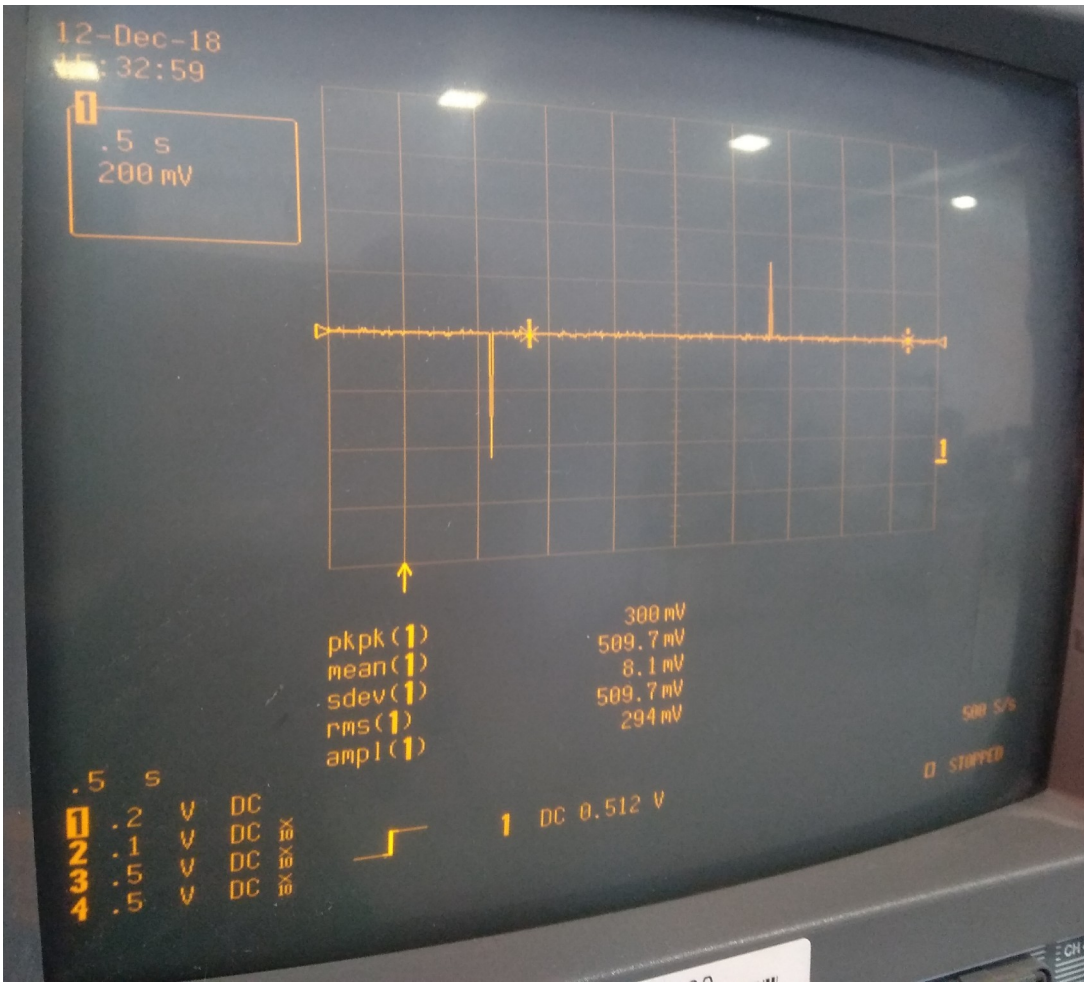


We reconnect the crystal again, (and for the rest of the tutorial the crystal will be connected). And first we check:
Signals from DTR, before C4: (black dot is where the oscilloscope is connected):



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



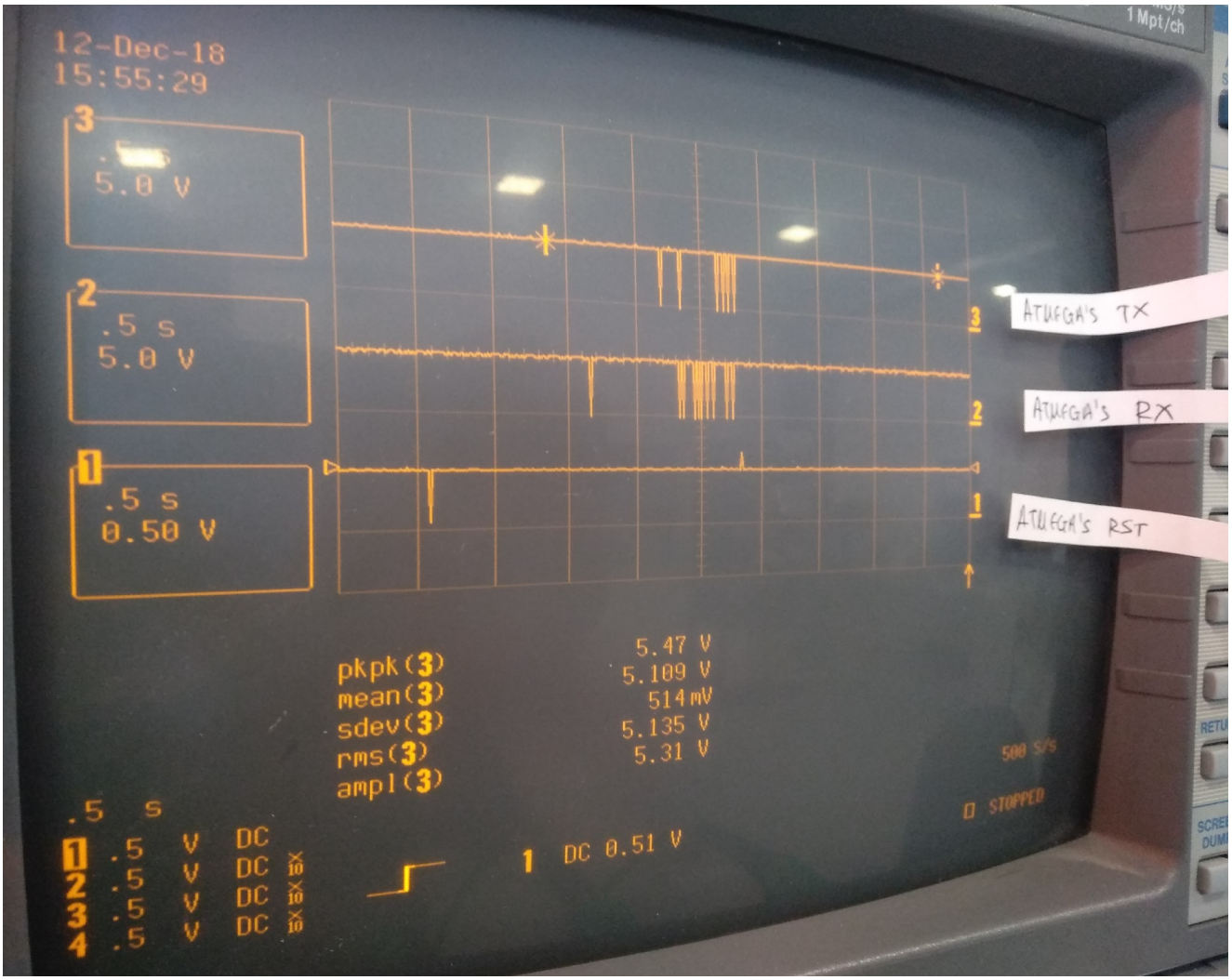


This is what the atmega sees.

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: 4/3/2019

By: Christianidis Vasileios
Contact info: basilisvirus@hotmail.com