Some of my viewers asked me « how can I use the Nextion on an ESP8266 or on an Arduino Uno”? Watchers of my videos #56 and #59 know, that I used an Arduino Mega for my tests, because this board has more than one Serial connection. So, I was able to connect my Nextion to Serial2 and use Serial for debugging. Very simple.

Other Arduinos, including the Uno only have one serial and the ESP8266 has 2 serials, but one is not completely usable. In this video, I will show you how to use a Nextion display also on these platforms.

Let’s first start with the Nextion Library. It supports two Serial connections, one for the Nextion itself, and one for debugging. The definition of the two connections is done in the file nexConfig.h in the libraries directory. The name of the library directory usually is something like: …\Arduino\libraries\ITEADLIB\_Arduino\_Nextion-master. Here you can open the file nexConfig.h with your text editor. For this video, you find three important defines:

1. #define DEBUG\_SERIAL\_ENABLE
2. #define nexSerial
3. #define dbSerial

For the Arduino Mega we defined DEBUG\_SERIAL\_ENABLE, set nexSerial to Serial2 and dbSerial to Serial. The Nextion was then connected to Serial2 and we were able to see the debug messages in serial monitor

If you do not define DEBUG\_SERIAL\_ENABLE or comment it, you will not get any debug messages and can live with one Serial connection. I show you this concept with an Arduino Uno.

The nexConfig.h looks like this: We comment the #define DEBUG\_SERIAL\_ENABLE and set the #define nexSerial to Serial. The #define dbSerial does not matter, because DEBUG\_SERIAL\_ENABLE is not defined. To make sure we create no problems later on, we define it to a Serial which is not available on this board. For the Arduino Uno, this is Serial 1 or 2.

In our sketch, we also have to comment all dbSerial statements. Because we defined dbSerial to an unavailable Serial connection, the compiler helps us in finding these statements, because we get errors if we forget to comment one of these statements.

For our tests I use the example of video #59. I connect the Nextion display to the RX and TX pins and pay attention, that I connect RX of the Arduino to the TX of the Nextion and vice versa. Now, I try to upload the sketch. It compiles, but unfortunately, does not upload. I only get error messages. Why is this?

The heart of our Arduino UNO is an Atmega328p chip. Because this chip does not have an USB connection, the Uno board contains an USB to Serial chip. This chip is connected to the two serial pins RX and TX. If we connect our Nextion display to the Uno, we have to use the same pins, because the 328p only offers this single serial connection. Now, we have two conflicting information coming to the same pin and this screws the whole communication up. The simple solution to this problem is to disconnect the Nextion during upload.

Now it works perfectly.

After the upload, we reconnect the Nextion display. Because no signal is coming from the USB connection, the display works now as expected. If we look to the serial monitor, we even see the raw messages coming from the display.

The same scenario works also with the ESP8266. Here, we also connect the Nextion to the RX and TX pins. We also have to disconnect the Nextion for upload.

This is what I call the “productive version” which I use for my devices if they are debugged and are used in a “productive environment”. In this scenario, I do not want to create any debug messages.

But if our sketch does not work yet and we need some debugging information? In this case, we have to find a way to get a second serial connection for our debugging information.

For Arduinos it is possible to create a second serial connection using the softwareSerial library. With this library, you can define two other pins as RX and TX for debugging. Unfortunately, I was not able to get SoftwareSerial working with the Nextion library. So, the only possibility I found was to disable the debug function in the Nextion library and create a new serial connection directly in our sketch and put it on pin 2 and 3. I call this connection “debugSerial”. Now, we can print some debug messages.

To display the messages, we have to connect pin 3 to a serial adapter and install a small program like PUTTY on our computer. Now, we can see the debug messages. In addition, we still can see the raw messages coming from the Nextion in our serial monitor. With this setup we can debug our sketch.

How can we connect the Nextion to the ESP? The ESP8266, like the Arduino Mega, has 2 serial connections. Unfortunately, one pin of the second connection cannot be used because it is connected internally to the flash ram. But good for us, it is the RX pin which is not usable. So, we can still use the TX pin to produce our debug messages.

The basic setup is the same as with the Arduino Uno. We connect the Nextion to Serial. For the debug connection we use Serial1. The TX1 pin is either GPIO2 on the ESP-12 modules or D4 on the NodeMCU boards. And again, we have to disconnect the Nextion display during upload.

The nexConfig.h file looks like this: We keep the #define DEBUG\_SERIAL\_ENABLE and set the nexSerial to Serial. The dbSerial is now set to Serial1.

Using this method, we can now use the Nextion displays on all Arduinos and also on the ESP8266.

Thanks for watching. I hope, this video was useful or at least interesting for you. Bye