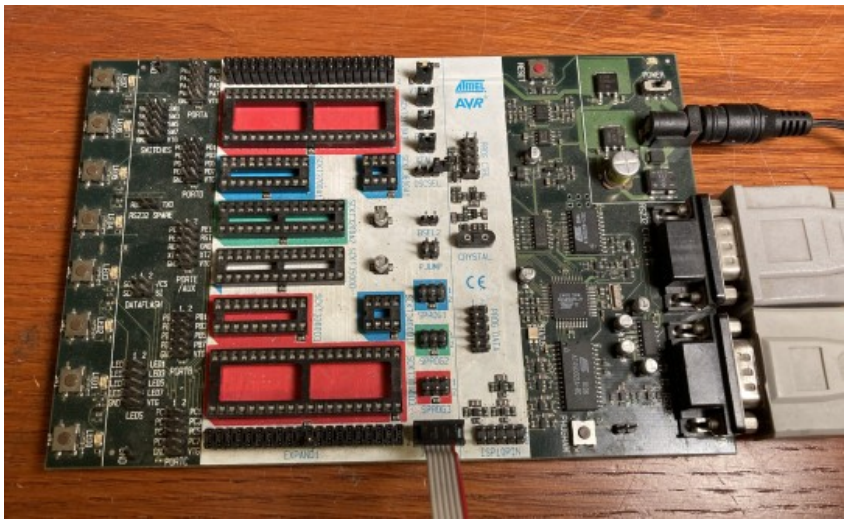
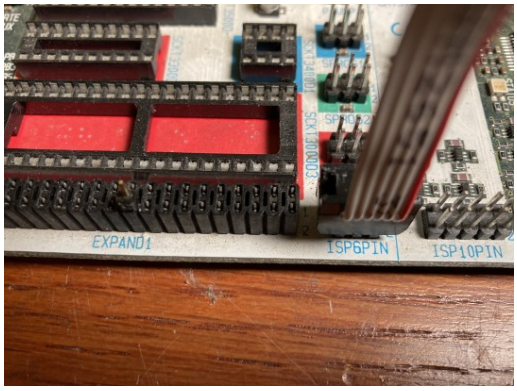
Instructions to setup Arduino UNO rev3 for ErgWare

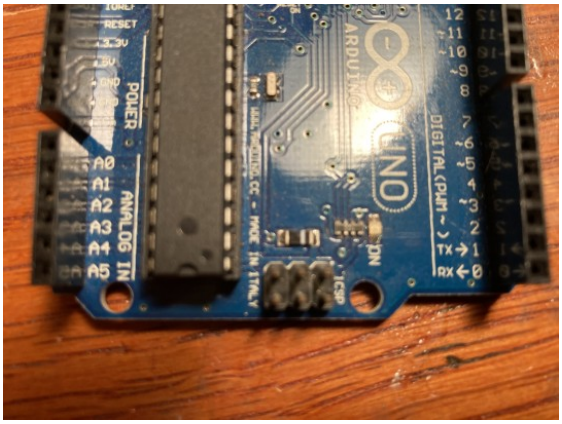
Step 1. Identify your programming board. I use the stk500. You might have the usbasp, or ponyprog or something else. The important thing is that it has to have an ISP programming interface. Some of these programmers are 10-pin, some are 6-pin. If you have the 10-pin type you'll have to get or make an adapter to 6-pin because the arduino uses 6-pin. My programming board has both.



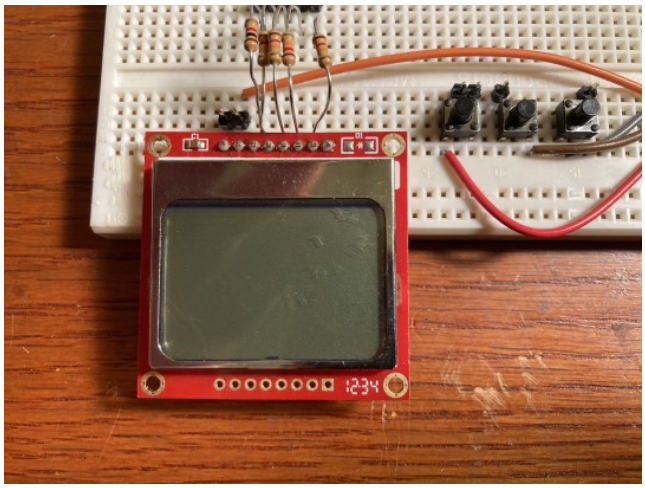


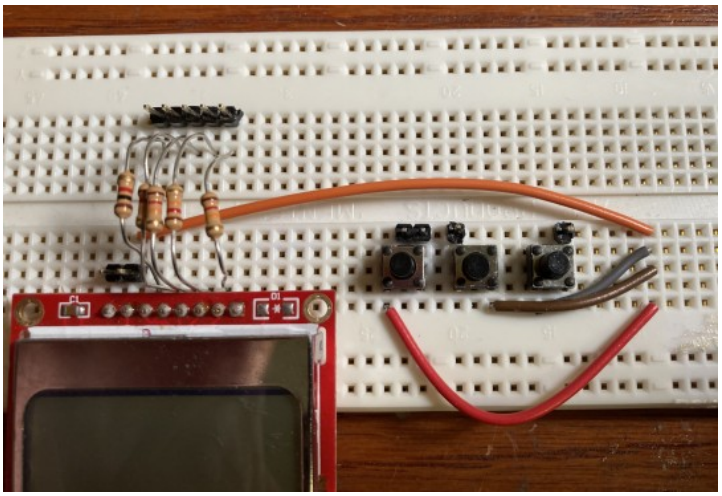
Step 2. Go get your Arduino.

Step 3. Identify the right ISP header on your Arduino. It is the one by the end of the board. It was labeled ICSP on mine.

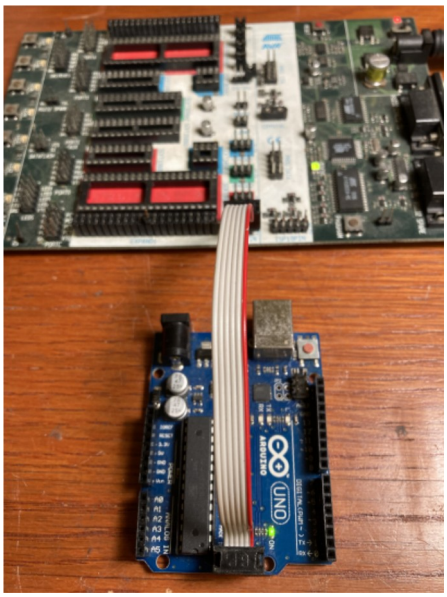


Step 4. Get other parts and breadboard them. The Nokia 5110 LCD and 3 switches. Note the **Nokia 5110 LCD must be powered (Vcc) off of the 3.3V** on the Arduino. I am showing resistors in-line for the other pins, **but in the end I did not use them**. You don't need them, though they might protect the screen a bit longer. You can actually do this step whenever you want before step 13 since we aren't yet going to hook anything up to the Arduino.

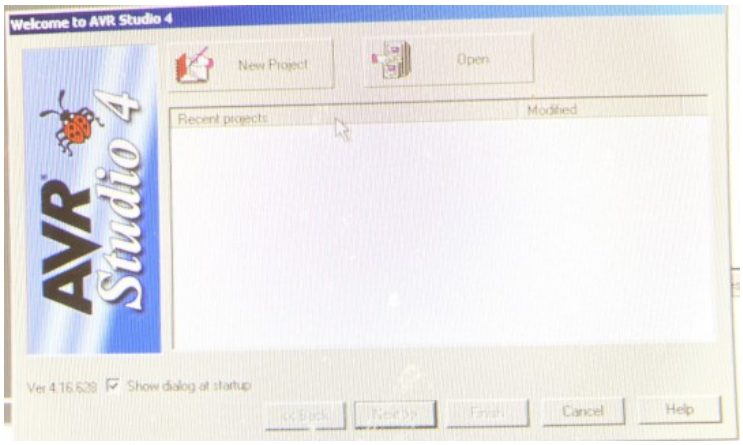




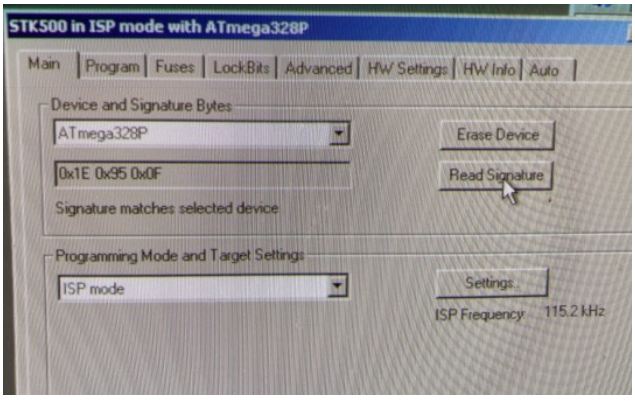
Step 5. Hook up the isp conector on the arduino to your programmer. Note, you have to get it the right way around or it won't work. You will find out soon if you have it right. Here is what mine looked like. **Note that I am not using any external power to the Arduino.** No USB. No power cable. Nothing. All of the power is coming in over the ISP cable.

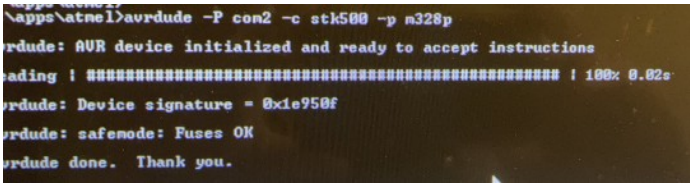


Step 6. Open up your programming environment. **You will probably NOT be using the standard Arduino environment you might be used to.** I use Atmel Studio 4, v 4.16.628. It is very old but in works for me. You can also just use the avrdude command line. You can also use programs like AVRDUDESS. All of these will work and will allow you to access the core device which is what we want to do.

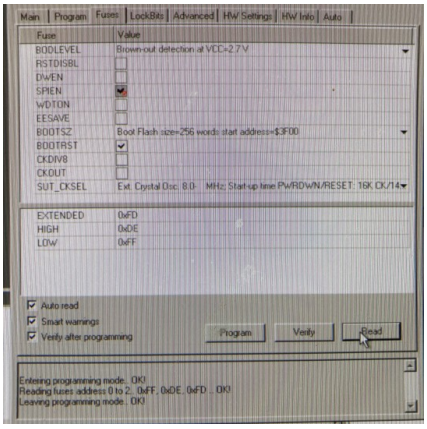


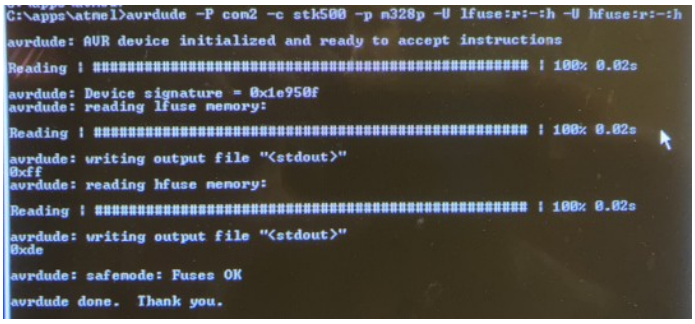
Step 7. The first really important step. Try to read the device signature. Below you can see screenshots of success in Atmel Studio 4 and on the avrdude command line. **Pay special attention to what you write on the avrdude command line. If you just type something randomly that you found on the web you might screw things up & "brick" your device.** The signature is 0x1E950F of this device. If you don't get this to work, **STOP AND FIGURE IT OUT.** Something is wrong either with your programmer, or your wiring or something. This step needs to work, don't try to force something with -F or anything in avrdude.



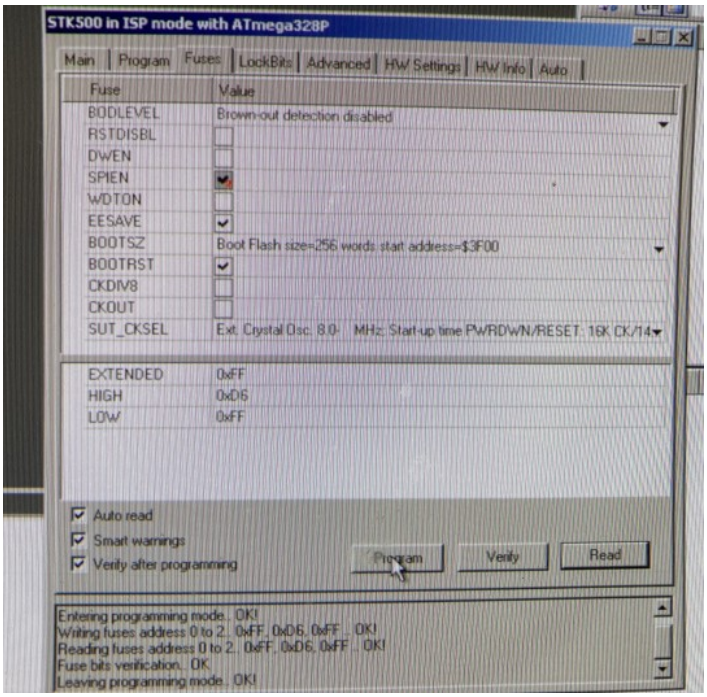


Step 8. We want to read the fuses (do **not** write them). This is just to ensure they can be read. My high fuse (Hfuse) read 0xDE, my low fuse (Lfuse) read 0xFF and my Extended fuse read 0xFD. This is what an "out of the box" Arduino reads, I think.



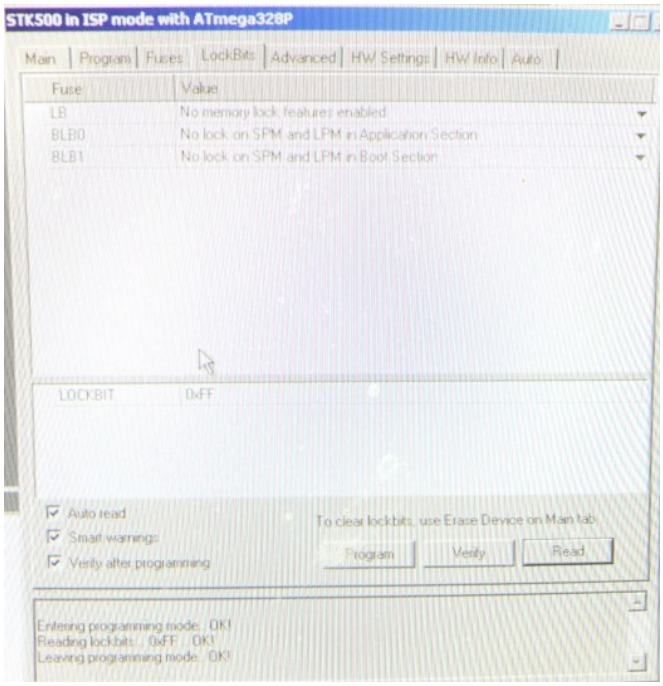


Step 9. We are now (sort of) at the point of no return. After steps 9 and 10 your Arduino will no longer function "like an Arduino". What this means is we are going to "overwrite" the Arduino bootloader so it will no longer work with the standard Arduino programming interface. There is a way to recover the bootloader if you want, but I won't go into that now. We are also going to set the EESave fuse since we will be using/saving the Eeprom. I also disabled brown-out detection. Now we need to program the fuses. Double, triple, quadruple check at this point before you hit "program" in your program or "enter" on the avrdude command line. The fuse settings need to be:
Extended: 0xFF
High: 0xD6
Low: 0xFF



Step 10. We also need to reprogram the “lockbits”. In Atmel Studio it says “to Clear Lockbits, use Erase device on Main tab”. So I went to the “main” tab and erased the device. It worked fine. There is also an avrdude command to erase the device
For my setup it would be:
avrdude -P com2 -c stk500 -p m328p -e

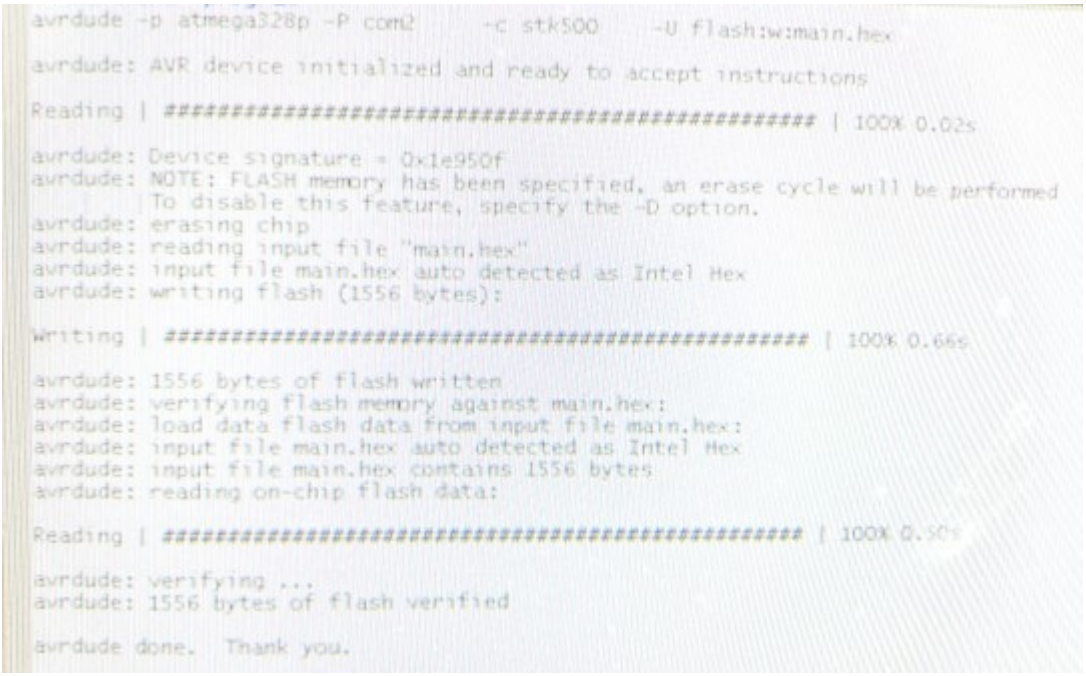
Yours might look a bit different in the -P and -c parts, but the -p mp328p and -e should be the same. I am sure other programs have similar capability. The final lockbit setting we want is **0xFF.** After this is done, the arduino board is now ready to be programmed.



Step 11. The next step is to fill up the EEPROM with the fonts. We are going to flash some code to flash memory to run this instead of just flashing the eeprom. Why? Only because I thought some people might want to change the code themselves so I gave the source code as well for the font table. In any case, go to the github

https://github.com/dvernooy/ErgWare/tree/master/v0.4

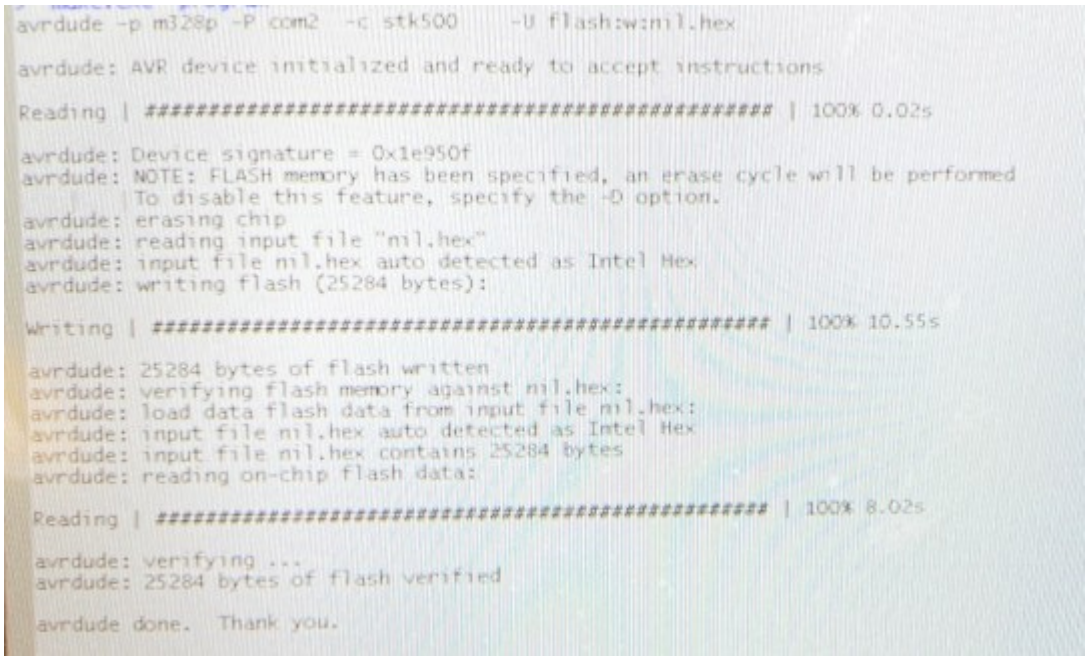
Find “main.hex” in the “eeprom_write” folder & flash it to the chip. **To be crystal clear, even though the purpose of this program “main.hex” is to fill up the EEPROM, you do not want to write main.hex to the EEPROM, you want to write main.hex to flash memory.** Below is what it looks like from the avrdude commands. In Atmel studio 4 you can also program the flash with a *.hex file on the program tab. Either way will work. **The final step to do at this point is to unpower the board, and then power it up again (or just hit the reset button on the Arduino) to ensure the EEPROM is now written.**



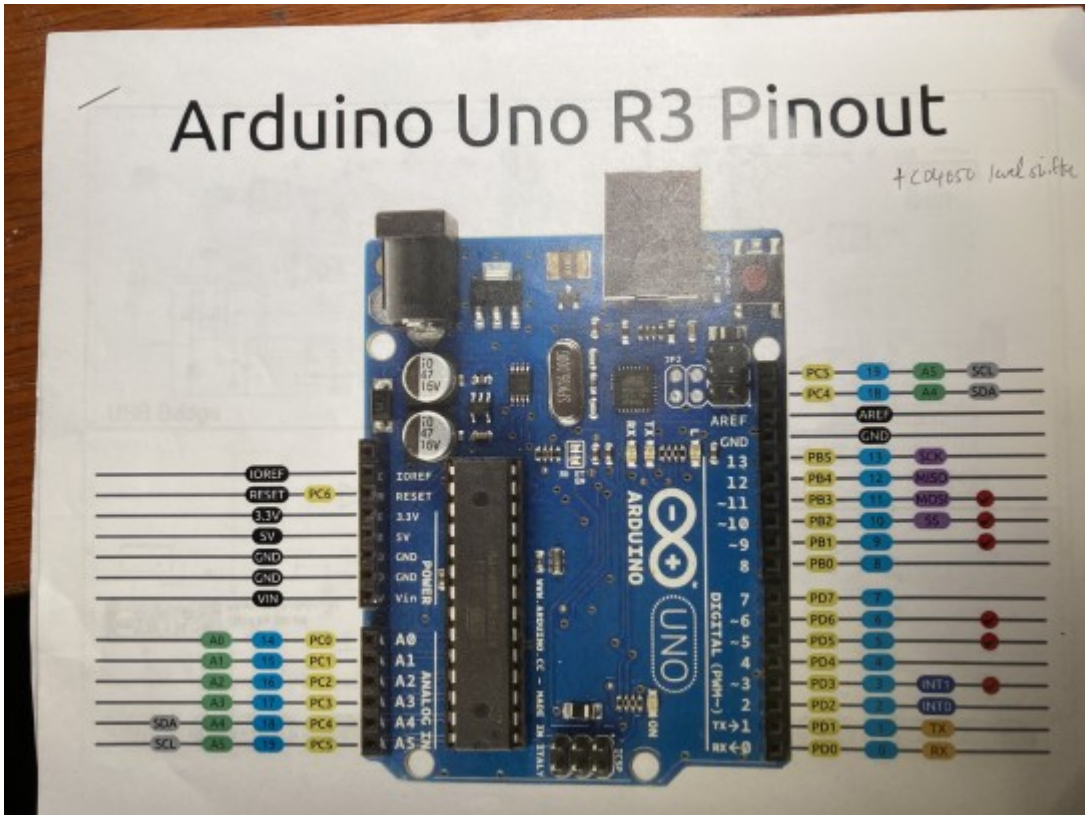
Step 12. We are now ready to flash the final program. Go to the github

https://github.com/dvernooy/ErgWare/tree/master/v0.4

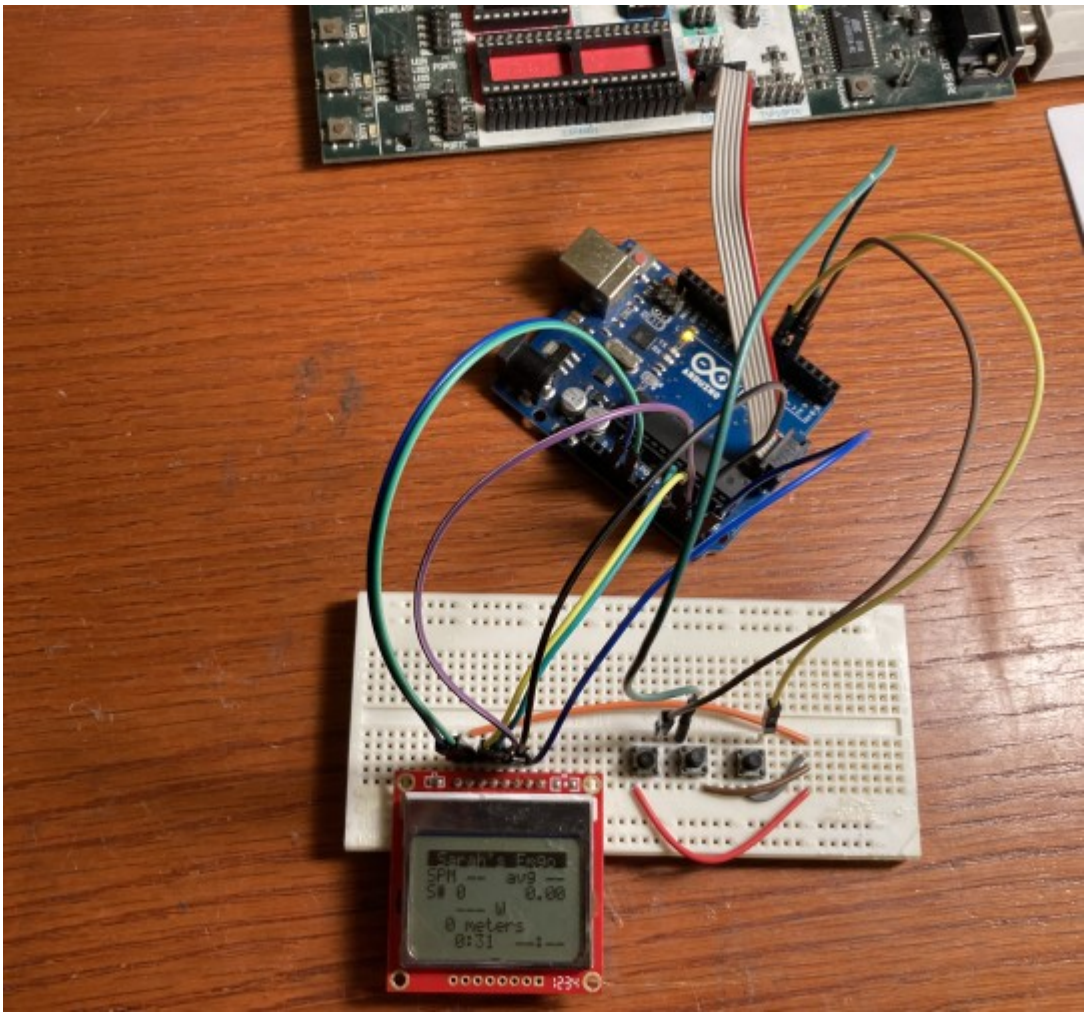
Find “nil.hex” in the “source” folder & flash it to the chip. Use a similar method as Step 11. You can see the avrdude command below.



Step 13. Time to wire up the board fully. Turn off the power (unplug the programming cable) while you do this. You can use this pinout diagram I found on the web that was really useful for me.



Step 14. Once it is all wired up correctly, you can re-apply power and you should see something on the LCD. You can press any button to escape from the splash screen and you should see the screen below. You can press the buttons to move around. **Note, at this point you can use anything to power the board … the programming cable, the usb port or the external power port.** They will all work!!!



Step 15. The only thing I have not yet wired up and tested is the photodiode/interrupter circuit with the chopper wheel. I am sure it will work. If there are problems let me know & I will help you.

Ping me if you have questions.

-Dave