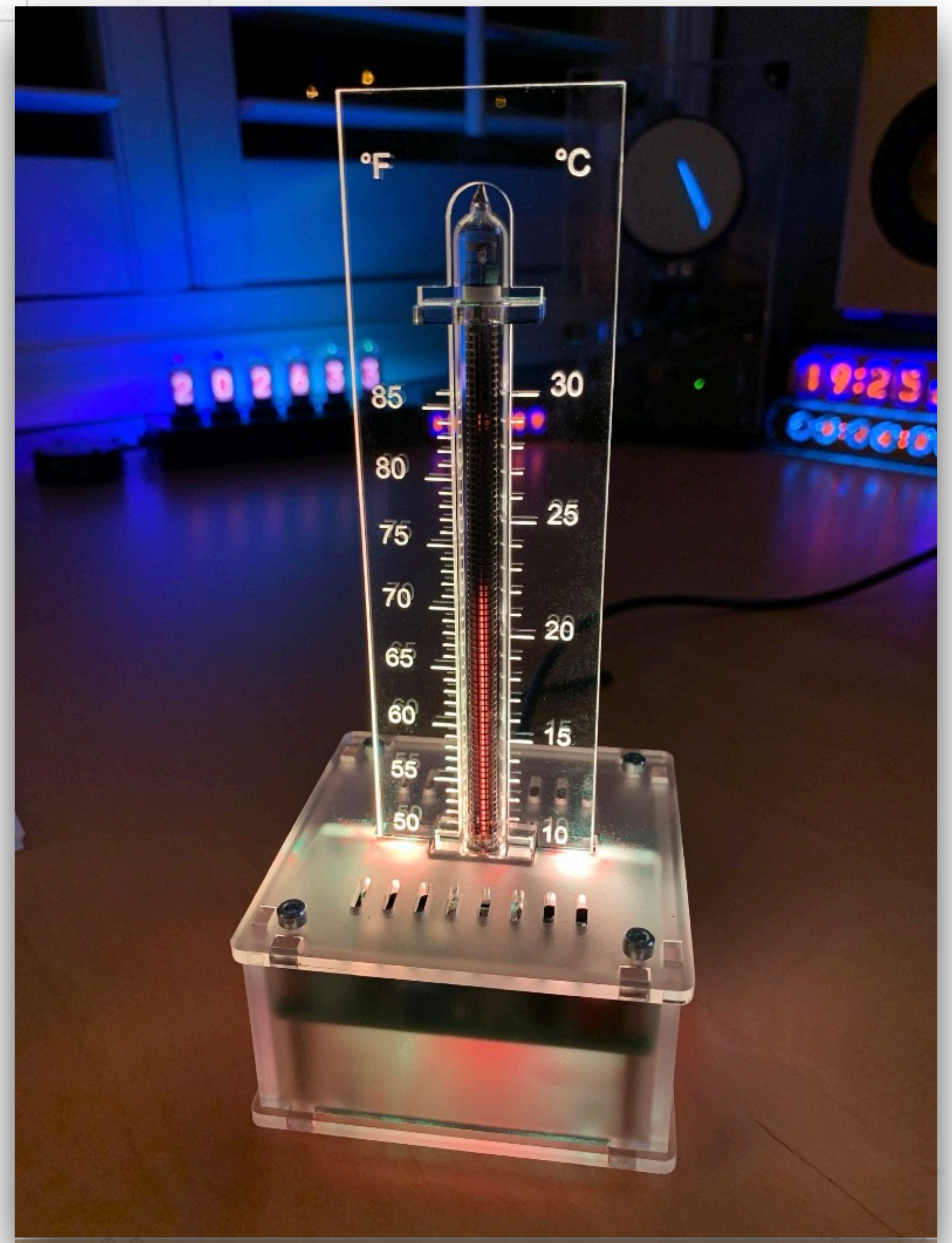
Nixie IN9 Bargraph Thermometer Assembly Manual



Revision 3 - 2021

Overview

This electronics kit is designed for hobbyists with intermediate experience in assembling electronics and is most definitely not for beginners. It requires a knowledge of soldering, programming using the Arduino[1] platform and a good grasp of basic electronics. If this is not you, then please do not attempt to build the kit unless you have qualified assistance.[2]

Please take your time – it will take approximately 2 hours to complete this kit if done correctly. It helps to have a clean, tidy work area and quality tools to complete the process.

Assemble the parts in the order as stated in the instructions – Please read and understand each step before you perform each operation for the best chance of success. If you have any questions at any point, feel free to contact us via email.[3]

The following tools and materials will be required to assemble the thermometer:

- A good quality soldering iron (25-40W) with a small tip (2-3 mm).
- Thin solder wire with no-clean flux. Do not use any flux or grease. Do not use lead-free solder.
- A set of small screwdrivers.
- A small wire cutter for electronics.
- Long nose pliers.
- A hot air soldering station / heat gun / lighter (or alternative).
- A multimeter
- A computer with the Arduino IDE installed.
- Solderwick (just in case….)

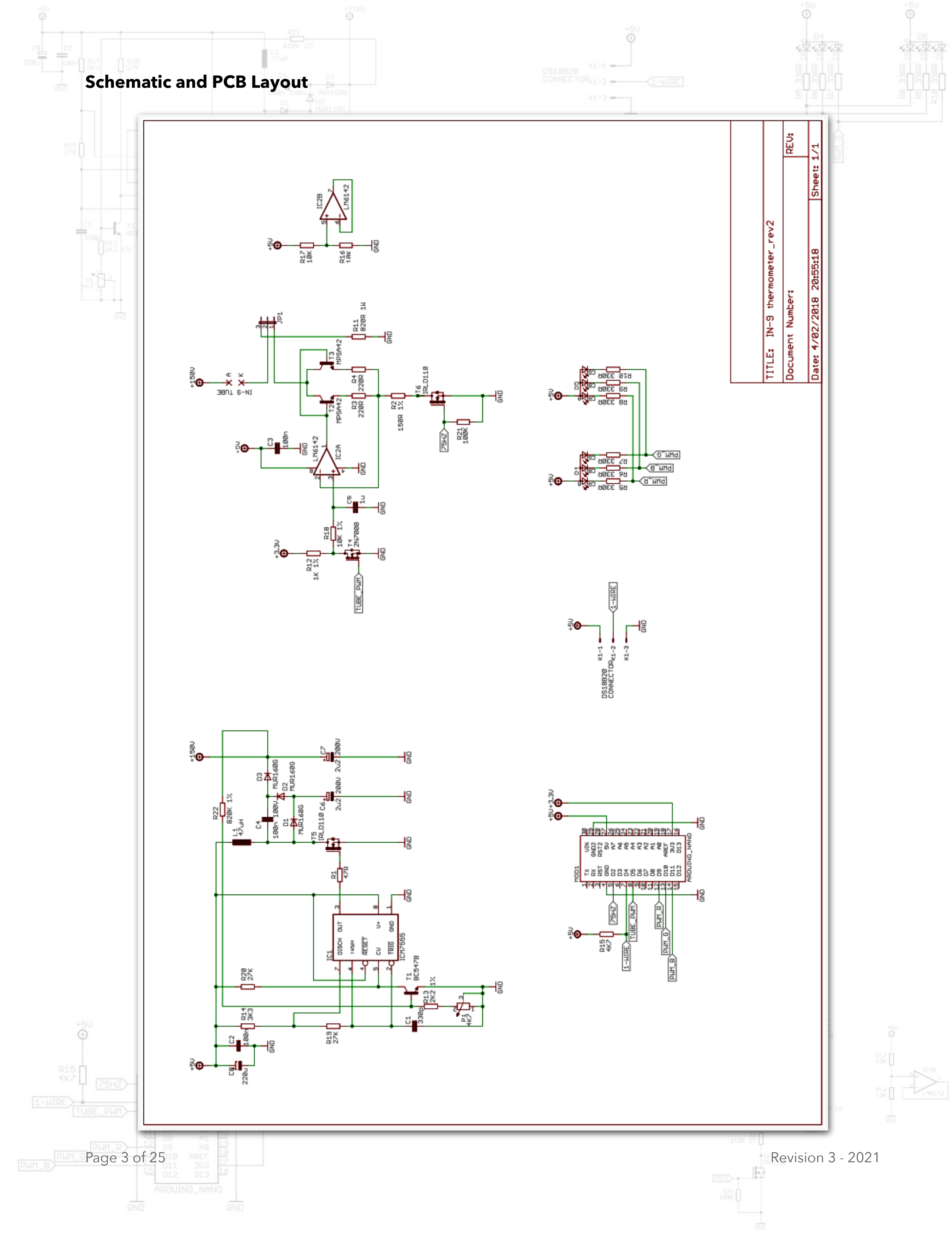
It is a good idea to read over the whole manual first to get a grasp of what is involved in the build process. Of note, it is imperative that every resistor is measured using a multimeter *prior* to being soldered into the boards. Tracing incorrect resistor placement can be quite challenging….[4]

CAUTION! The approximately 150V voltage generated by the thermometer circuitry is potentially dangerous. Do not touch the circuit when it is in operation and use the thermometer only when mounted inside the supplied enclosure.

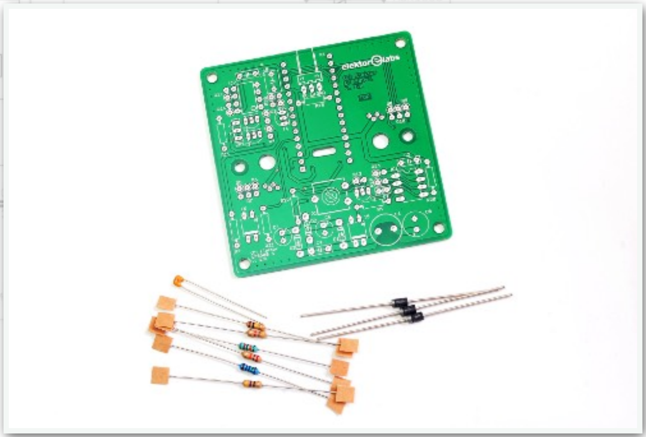
Note: This manual is written for thermometer software dated 2018-05-03.

[1] www.arduino.cc
[2] We don't want you to be disappointed.
[3] stocksclocks@gmail.com
[4] Ask me how I know…. :)

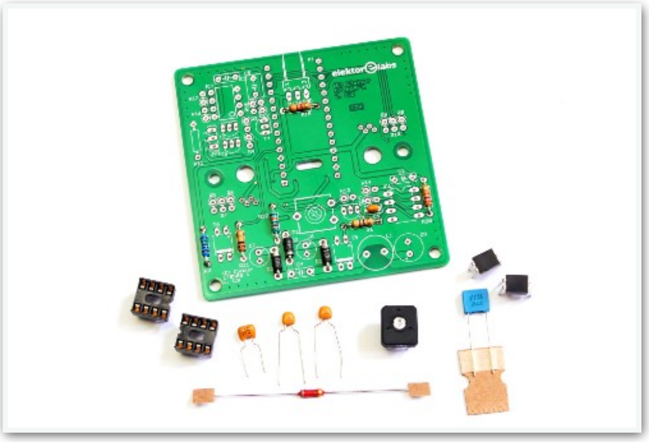
Schematic and PCB Layout



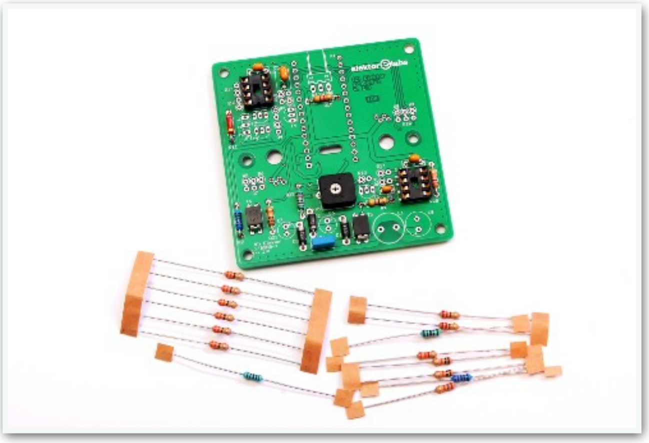
Assembling the Main PCB



Mount the components from low to high. Start with the horizontally mounted resistors R1 (47Ω), R2 (150Ω 1%), R15 (4K7), R20 (27K), R21 (100K) and R22 (820K 1%). Don't install R11 now despite the fact that it is also mounted horizontally. Mount the diodes D1, D2, D3 (MUR160G). Finally mount C1 (330pF). Reading the color bands of the 1% resistors can sometimes be difficult even if you have a very good eyesight. Always, check the value using a multimeter before mounting these resistors.



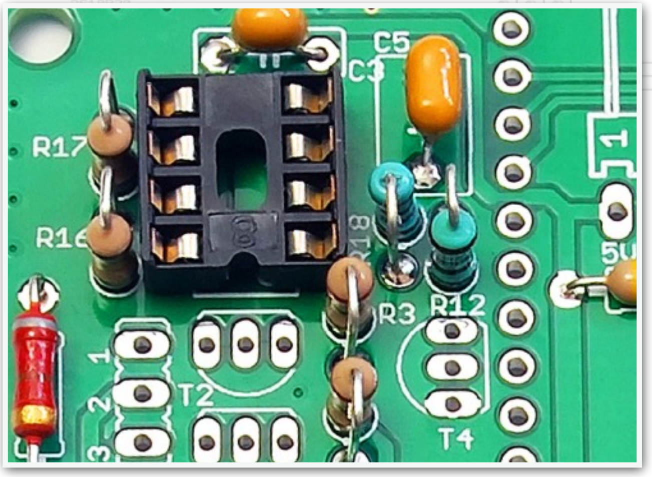
Mount the IC sockets for IC1 (8p) and IC2 (8p). Insert the trim potentiometer P1 (4K7), MLCC capacitors C2, C3 (100nF) and C5 (1µF). The trim potentiometer should be set in the middle position.



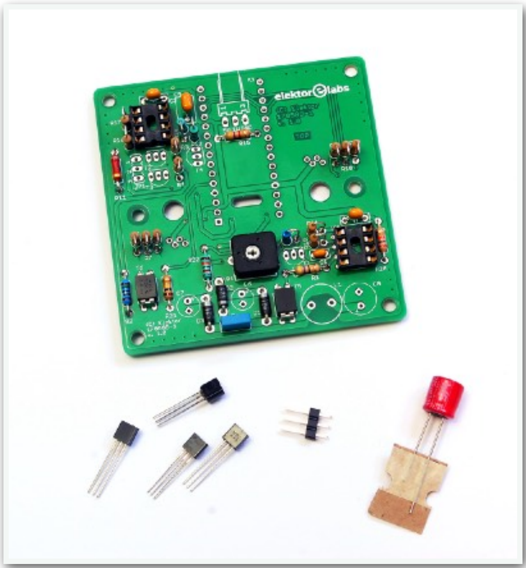
Mount the MOSFET transistors T5 and T6 (IRLD110), the side with the two leads connected together should match the dash marking on the PCB. Mount film capacitor C4 (100 nF 100 V). Finally mount power resistor R11 (820 Ω 1 W). Make sure there is a space of a few millimeters between the PCB and R11 as this component will get hot during a burn-in procedure of the IN-9 tube.

Mount the remaining resistors R3, R4 (220Ω), R5 ... R10 (330Ω), R12 (1K 1%), R13 (2K2 1%), R14 (3K3) R16, R17 (10K), R18 (10K 1%) and R19 (27K) in an upright position.

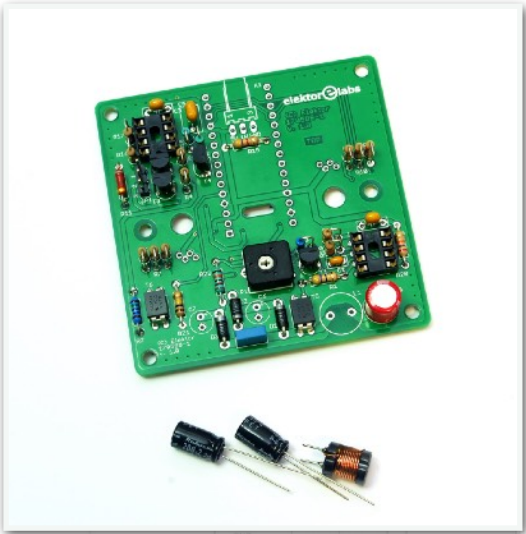
The solder pads for R12 and R18 form a 2 by 2 matrix. Make sure the resistors are oriented as shown in the picture to the right.



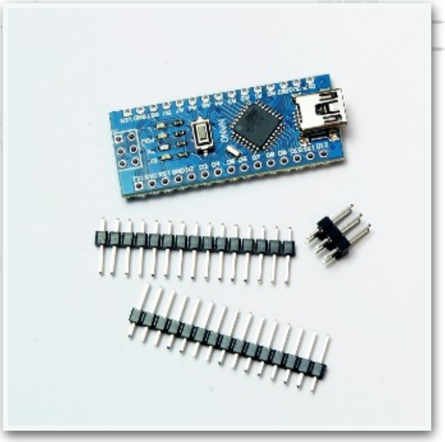
Mount the pin header JP1 (1x3p), transistor T1 (BC547B), transistors T2 and T3 (MPSA42) and MOSFET transistor T4 (2N7000). Finally insert and solder C8 (220µF).

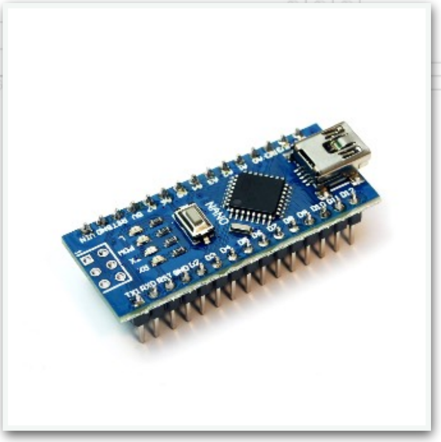


Mount the remaining larger components. Start with inductor L1 (47µH 1.2A), then mount capacitors C6 and C7 (2.2µF 200V).

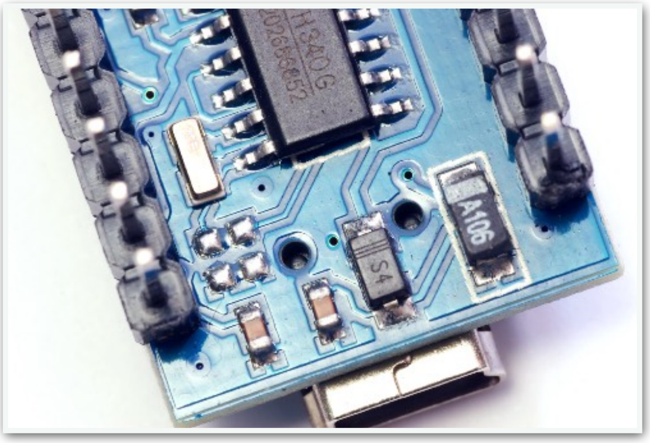


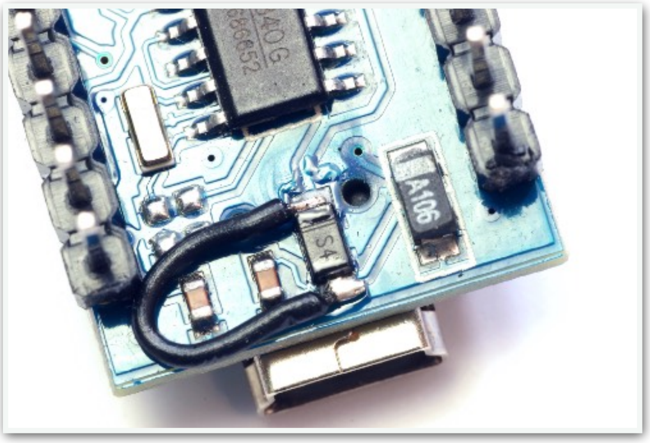
Your Arduino Nano may come with the pin headers already soldered in place. If not, mount the two 15-way pin headers to the Arduino as in the pictures. It's not necessary to mount the small 2x3 pin header (which is used for ISP programming). If the 6-pin header is already soldered to your Arduino Nano then you'll have to remove it first. The easiest way is to first slide the plastic base over the pins away from the board, then desolder the individual pins.





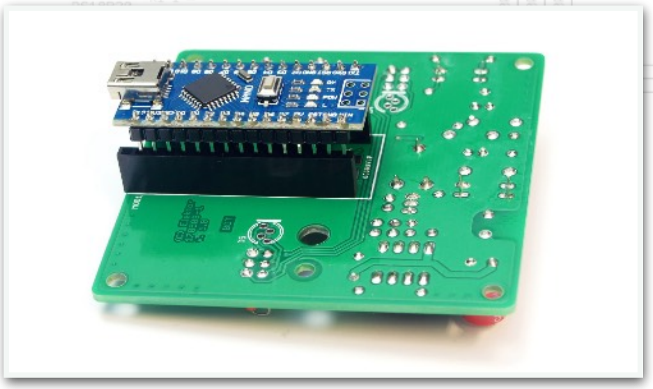
The Arduino Nano comes with a Schottky diode connected between the USB connector and the Arduino 5V power rail in order to prevent current flowing back to the USB interface when the Arduino is powered by an external power source. On Arduino Nano clones the specs of this diode are marginal and often a small signal diode rated for a few hundreds of milliamps is used. During a burn-in procedure of the IN-9 tube, the total current consumption of the circuit can go up to 700 – 800 mA with the diode becoming blisteringly hot. As we are not going to apply external power to the Arduino, the diode is not really useful and we recommend shorting it with a small piece of wire.



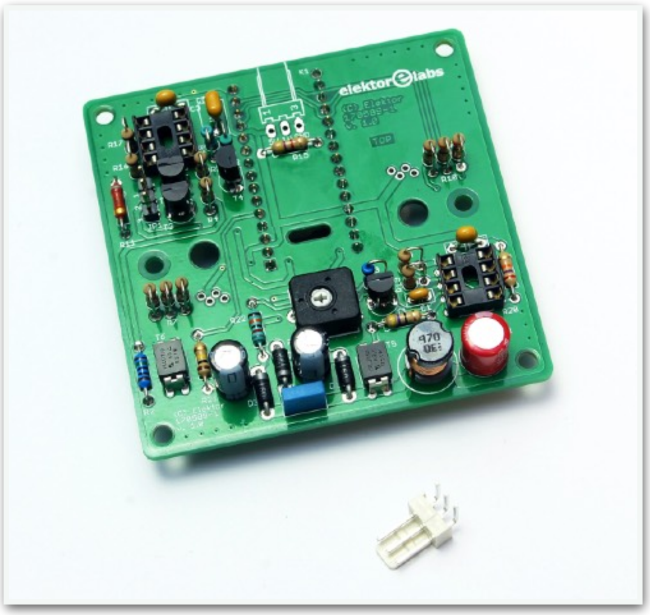


The socket headers delivered with this kit are 16-way while the pin headers on the Arduino Nano have only 15 pins. The latter socket headers are difficult to get so this means you will have to remove one contact pin from each socket header with long nose pliers/cutters which is easy to do.

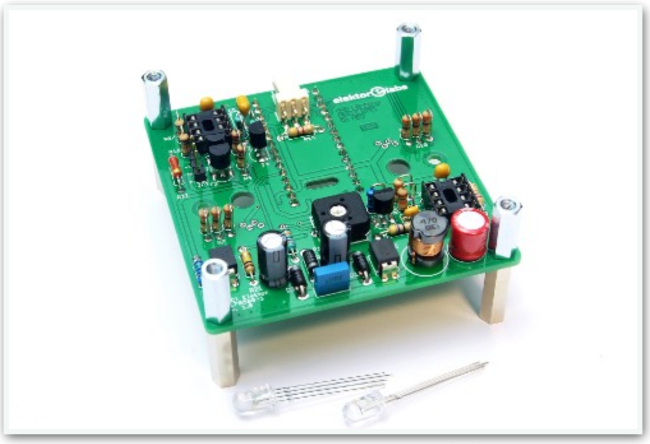
Mount the socket headers on the solder side with the side with the removed contact pin flush with the border of the PCB. Solder only two pins for each socket header so you still can correct their position if necessary. Fit the Arduino loosely into the socket headers to check for correct alignment. If everything is ok, solder the remaining pins of the socket headers.



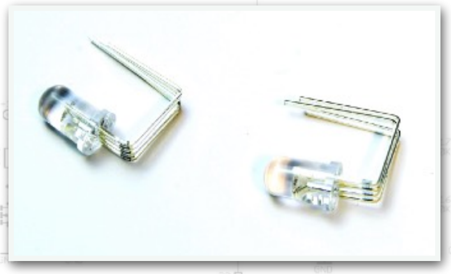
Finally mount locking pin header K1 (1 x 3p).



Take the 14mm M3 M/F and 20mm M3 F/F standoffs and mount them to the board as shown. This will make the following steps easier as the PCB can now rest on the standoffs.

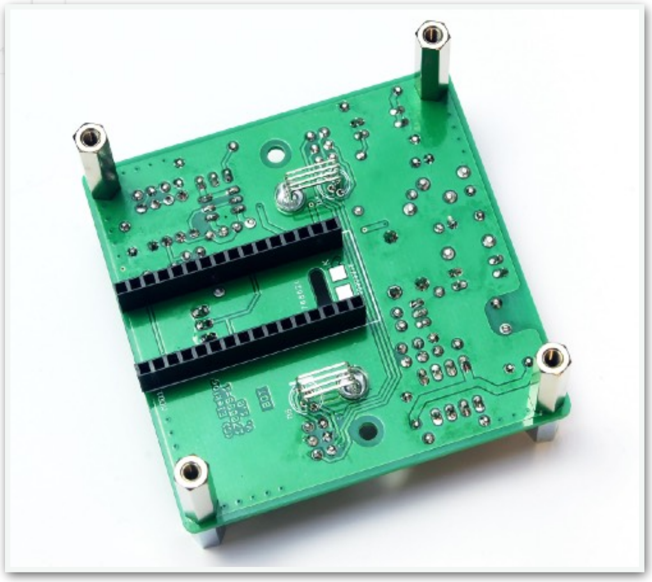


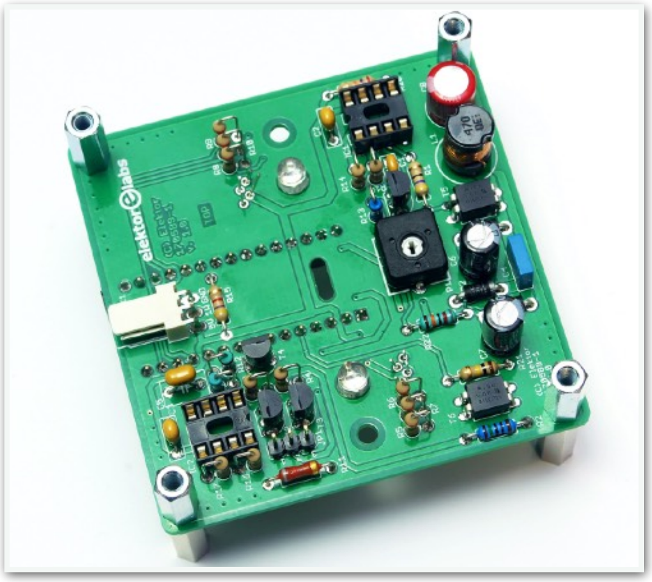
Bend the lead wires of the RGB LEDs in two 90 degree angles. Bend them first right above the stoppers in the lead wires using long nose pliers and put them through the 5 mm holes in the PCB from the solder side. Sometimes it might be necessary to enlarge the 5 mm holes slightly to get a good fit. Now, with the PCB as a guide, mark where you should apply the second 90 degree bend. Remove the LED from the PCB and apply the second 90 degree



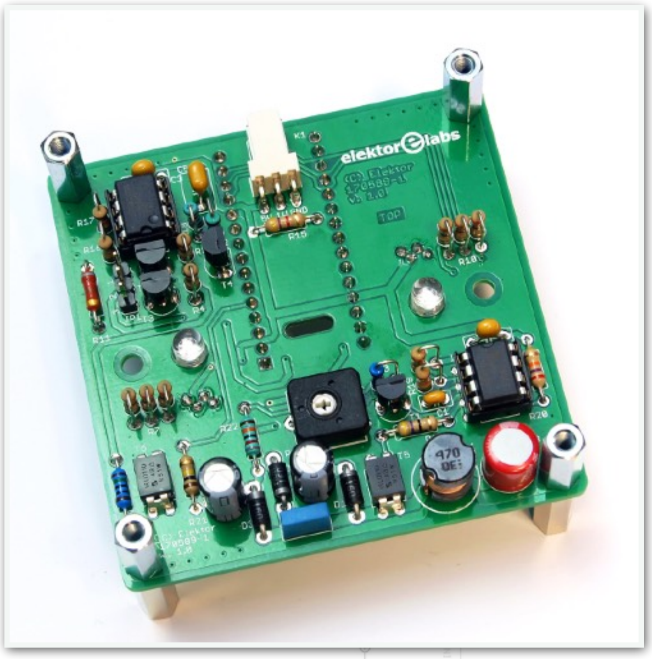
bend. Both RGB LEDs should be bent in opposite ways to each other.

Carefully guide the lead wires of the RGB LEDs through the solder pads on the PCB from the solder side. In the meantime push the RGB LEDs through the 5 mm holes so they protrude from the component side. Make sure the flat side is aligned with the dash markings on the PCB. Check if everything is positioned correctly and solder the leads from the component side. Do not apply too much heat on the RGB LEDs! Finally check if the leads of the RGB LEDs don't touch each other.



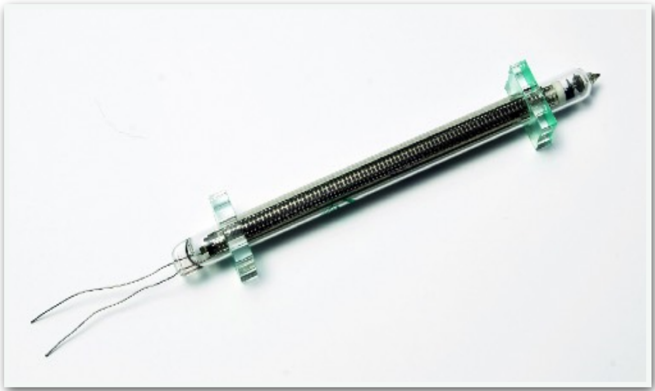


Lastly, put IC1 (ICM7555) and IC2 (LM6142) into their respective sockets. The PCB is now finished and ready for mounting the scale and the IN-9 tube! Take a break.



Assembling the Scale

There are quite some tolerances on the outside diameter of IN-9 tubes. To accommodate for these tolerances, the kit contains four sets of two tube holders with increasing diameters (0.25mm increments). Unpack the IN-9 tube and try out several of the tube holders to find out which pieces best fit the tube. Ideally the tube holder should not be so loose that the tube can slide out under its own weight but also not too tight as this may cause the tube to break.

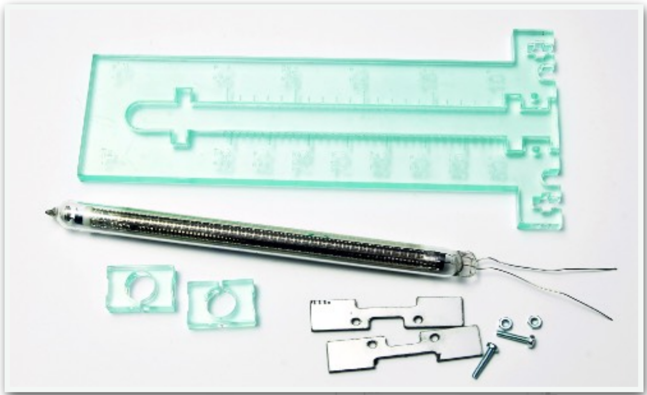


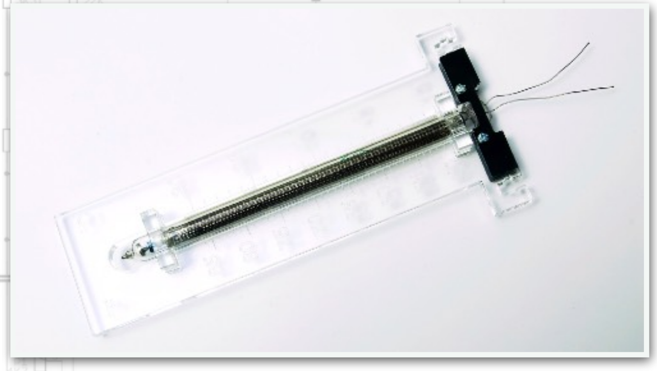
Slide the two tube holders you are going to use for the assembly a few times back and forth over the tube and rotate them a few times. This will make the final mounting of the tube into the scale easier. Do not discard the remaining tube holders as you may need them in case you need to replace the IN-9 tube.

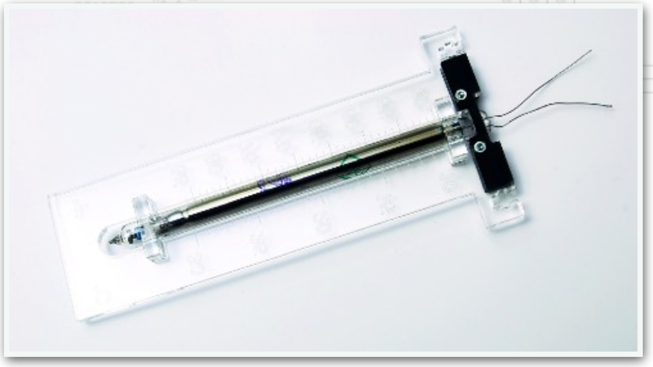
Remove the acrylic piece at the center of the scale and discard it. Also, remove the protective coatings from the tube holders and click them in place into the scale with their open end facing backwards.

Carefully slide the IN-9 tube into the tube holders from below. The anode grid should face the front side. Align the tube so that the light column will start just below the 10°C mark on the scale.

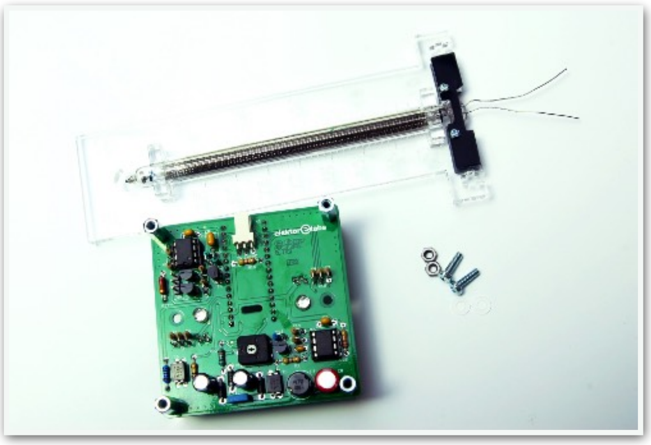
Remove the protective coating from the RGB led shielding. Only the metallic side has a protective coating. Now remove the protective coatings from the scale and attach the RGB led shields on both sides with the metallic sides facing inwards. Screw everything together with two M2x10 machine screws and two M2 nuts. The scale assembly is now finished.

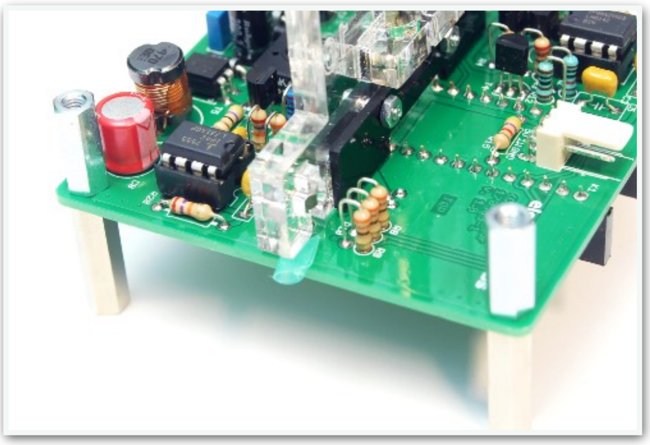


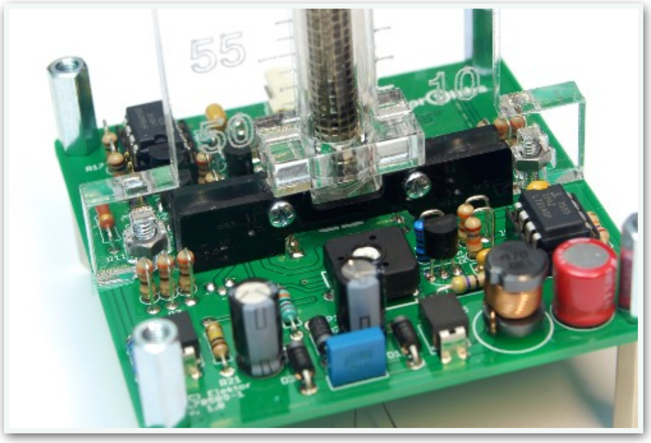


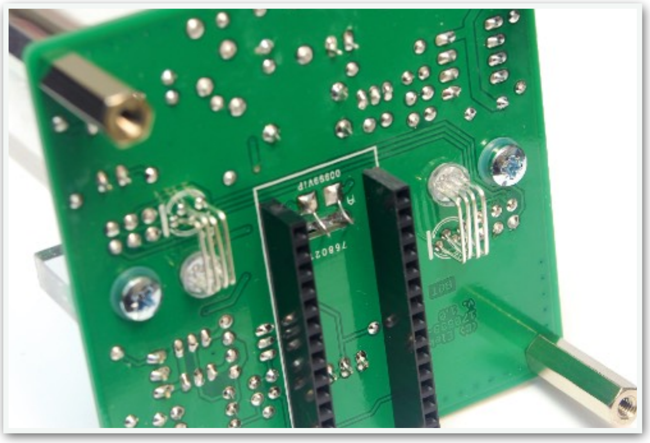


Carefully guide the leads of the IN-9 tube through the slotted hole in the PCB. Attach the scale assembly to the PCB using two M3x10 machine screws and two M3 nuts. Insert a plastic M3 washer between the PCB and the heads of the screws. Do not over-tighten the screws as the acrylic may break. Make sure the orientation (front/back) is correct.



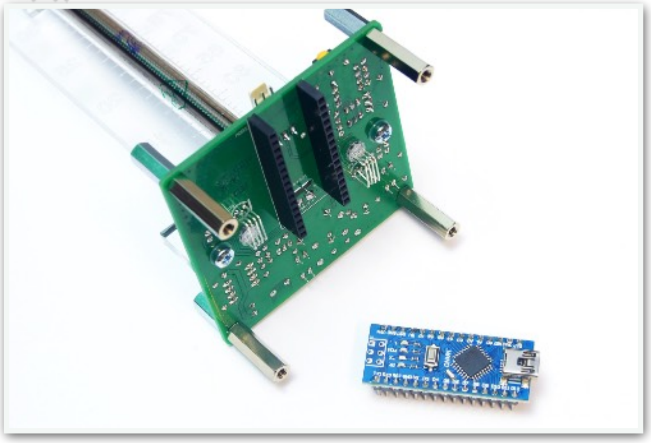


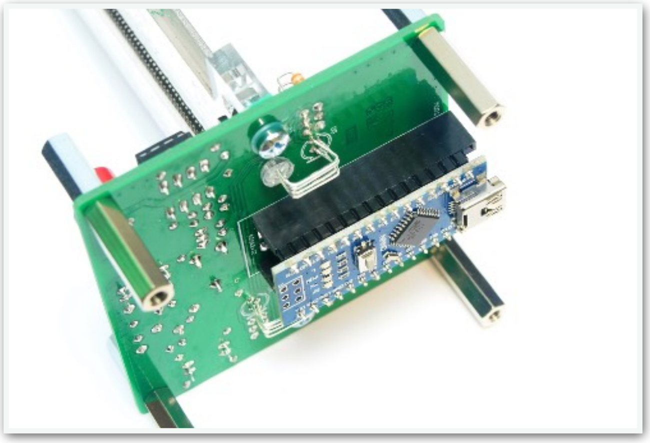




You may notice that the scale is not perfectly upright. This is caused by the fact that the acrylic may be a little warped during laser cutting. Also, the laser beam is not perfectly straight but widens a bit above and under the focal point of the focusing lens of the cutting head. As a result, the cut edges of the acrylic may not be perfectly 90 degrees with regards to the surface. This can easily be corrected by inserting a small piece of double or quadruple folded plastic or paper between the scale and the PCB. First loosen the screws on one side, slide the plastic or paper piece in place and tighten again. You can also use a small piece of the protective coating from the acrylic you removed during the previous steps for this purpose. Finally shorten the leads of the IN-9 tube and solder the ends to the solder pads marked with “A” and “K”. Be careful as the leads sometimes easily snap off at the spot where they enter the glass envelope of the tube. Make sure the leads are not subjected to any mechanical tension by bending them into a small arc before soldering them. Finally check if they are not touching each other or anything except for the epoxy material of the PCB.

Insert the Arduino Nano into the socket headers with the USB connector facing the backside of the PCB. Be careful to put it in the right way into the 16-way socket headers as the Arduino Nano has only 15-way pin headers.

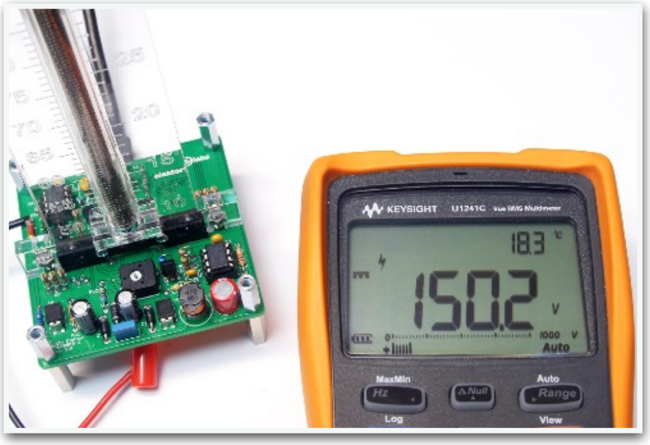


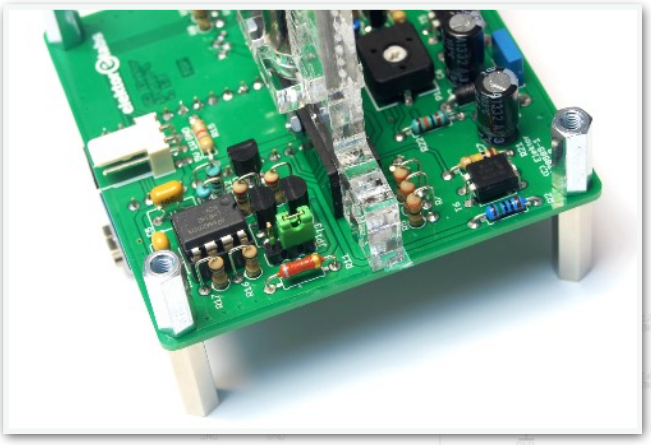


Make sure the IN-9 tube leads do not touch the Arduino Nano module in any way.

Power up the circuit using a 5V USB power supply. Except for some flickering LEDs on the Arduino Nano, nothing should happen.

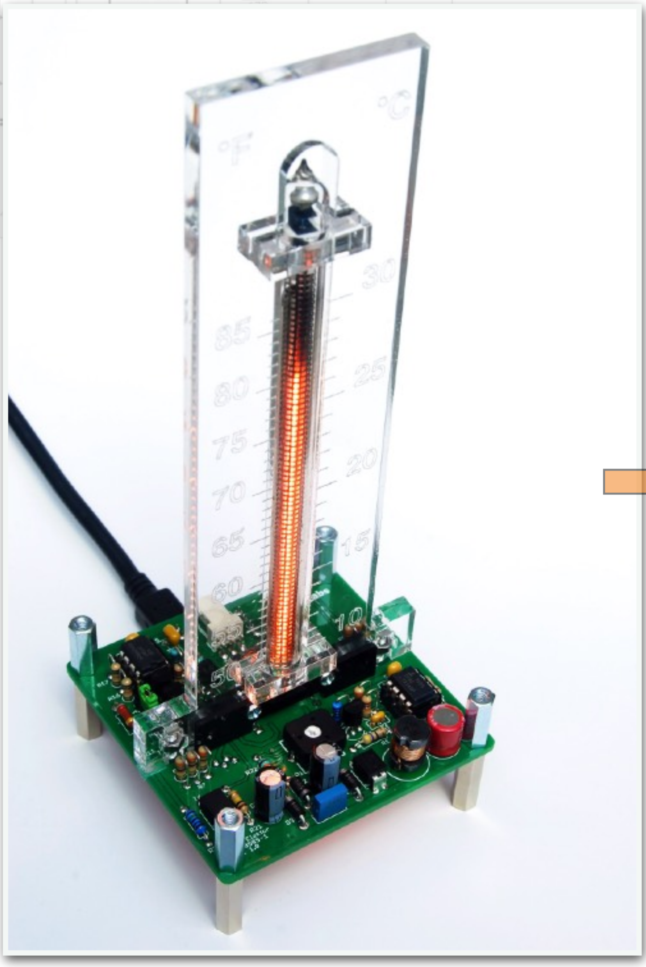
Now measure the anode voltage for the IN-9 tube. You can measure this between ground (GND) and the anode connection of the IN-9 tube or the cathode of D3. Choose whatever works best for you. Adjust the voltage to approximately 150V using potentiometer P1. If you are using an argon filled IN-9 tube, you can adjust the voltage to 160V. The IN-9 tubes supplied with the kits are neon filled tubes. If everything checks out, then disconnect the power again.

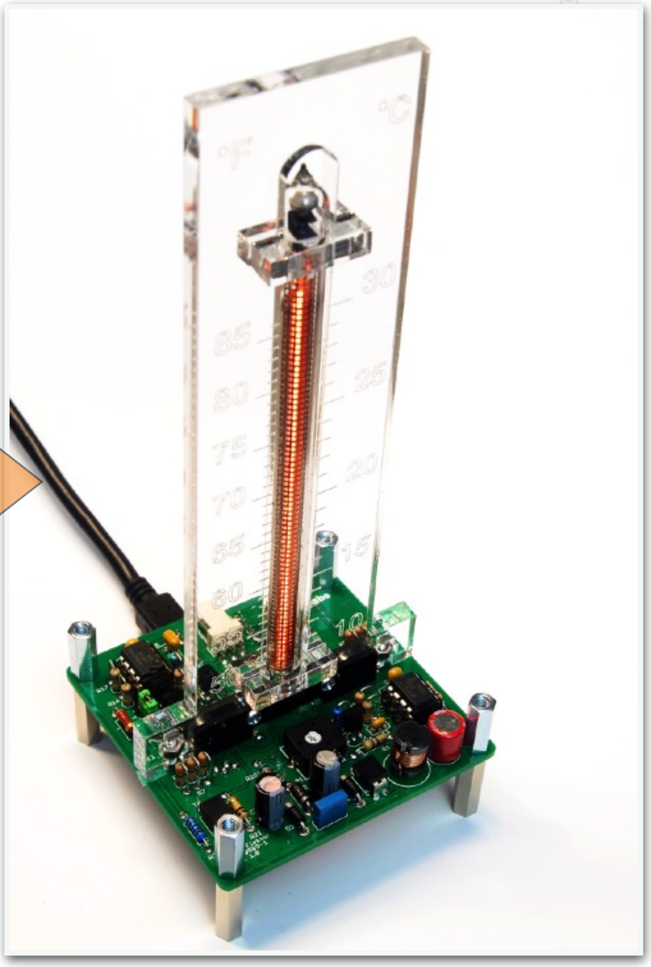




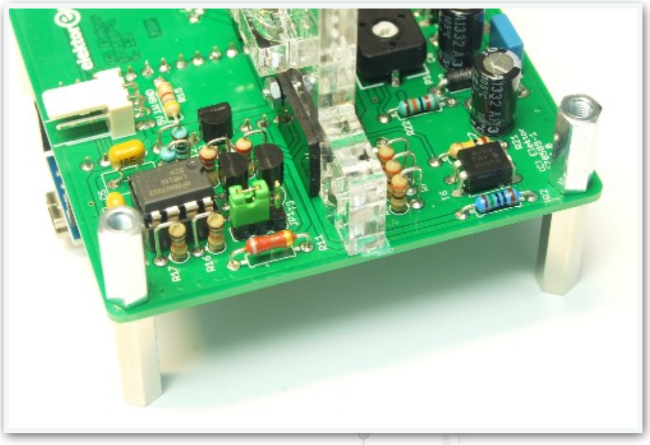
Connect pins 2 and 3 of JP1 with the supplied jumper. Power up the thermometer again using a USB power supply capable of supplying at least 1A. Do not connect the thermometer to a USB port of a computer!!!

Now the IN-9 tube should glow with a relatively high intensity. Wait until the glow reaches the maximum display height which is a few mm above the 30°C mark on the scale. This may take a while and it is normal that some components on the PCB and the IN-9 tube itself get hot.



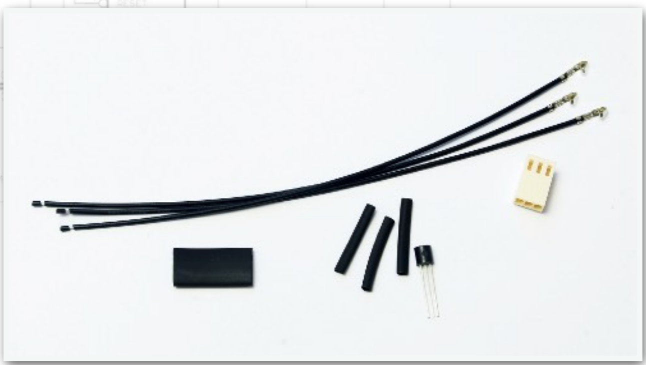


Disconnect the power and move the jumper to pins 1 and 2 of JP1.



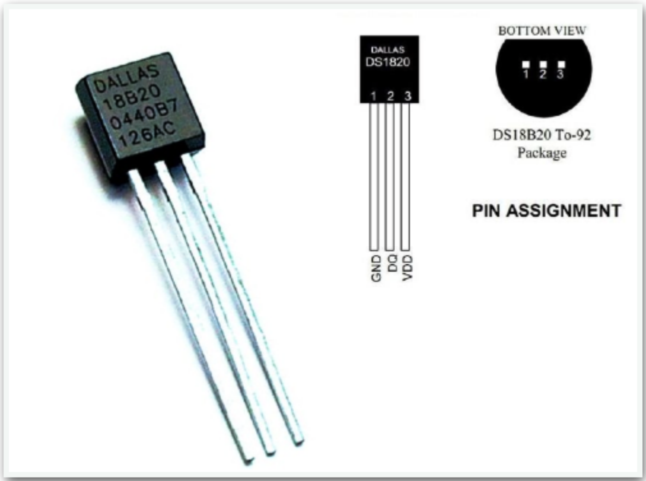
Assembling the Temperature Sensor

Slide the three pre-crimped wires into the 3-pole female connector from the backside until they snap into place.



Slide 2cm pieces of 3.2mm heat shrink tube over the wires and solder the wires to the DS18B20 sensor. The wire marked with "1" on the female connector is the +5 V power supply. The middle wire is DQ (1-Wire data). The remaining wire is ground (GND). Please take a look at the DS18B20 pinout in order to connect the wires correctly.

Activate the heat shrink tube using a heat source, slide a 2cm piece of 6.4mm heat shrink tube over the sensor and shrink again. The sensor is now ready to use.



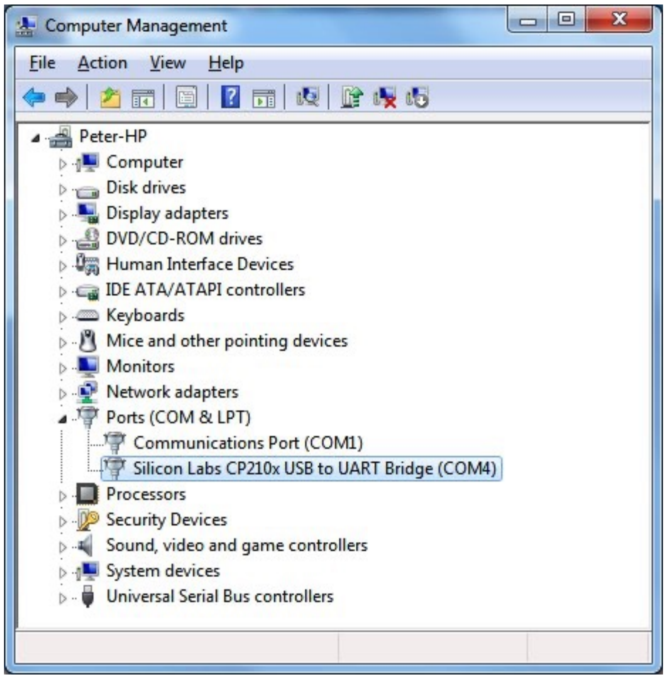
Connect the temperature sensor to K6 on the main PCB of the thermometer.

Setting up the Software

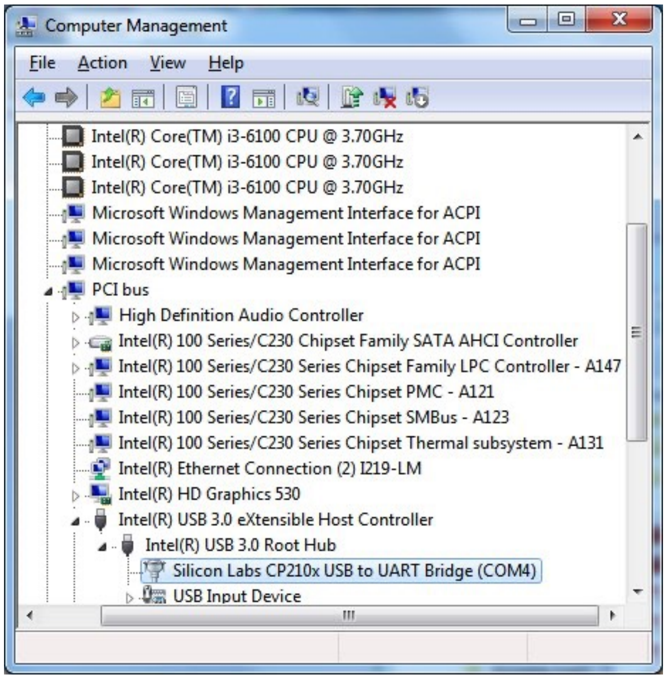
This section is for the most part aimed at using the Arduino IDE in Windows 7. You should have no problem when using another version of Windows or any other operating system that's supported by the Arduino IDE.

Serial Interface

Connect the Arduino Nano with a PC using a USB cable. The Arduino Nano incorporates a USB-to-serial function that enables the PC to access the thermometer as a serial interface.
If you're using Linux, the system will create an entry in the device directory structure e.g. /dev/ttyUSB0. Run command lsusb to obtain a list of all connected USB devices.



```
Peter-HP
  Computer
  Disk drives
  Display adapters
  DVD/CD-ROM drives
  Human Interface Devices
  IDE ATA/ATAPI controllers
  Keyboards
  Mice and other pointing devices
  Monitors
  Network adapters
  Ports (COM & LPT)
    Communications Port (COM1)
    Silicon Labs CP210x USB to UART Bridge (COM4)
  Processors
  Security Devices
  Sound, video and game controllers
  System devices
  Universal Serial Bus controllers
```



```
Intel(R) Core(TM) i3-6100 CPU @ 3.70GHz
Intel(R) Core(TM) i3-6100 CPU @ 3.70GHz
Intel(R) Core(TM) i3-6100 CPU @ 3.70GHz
Microsoft Windows Management Interface for ACPI
Microsoft Windows Management Interface for ACPI
Microsoft Windows Management Interface for ACPI
PCI bus
  High Definition Audio Controller
  Intel(R) 100 Series/C230 Chipset Family SATA AHCI Controller
  Intel(R) 100 Series/C230 Series Chipset Family LPC Controller - A147
  Intel(R) 100 Series/C230 Series Chipset PMC - A121
  Intel(R) 100 Series/C230 Series Chipset SMBus - A123
  Intel(R) 100 Series/C230 Series Chipset Thermal subsystem - A131
  Intel(R) Ethernet Connection (2) I219-LM
  Intel(R) HD Graphics 530
  Intel(R) USB 3.0 eXtensible Host Controller
    Intel(R) USB 3.0 Root Hub
      Silicon Labs CP210x USB to UART Bridge (COM4)
      USB Input Device
```

If you're using Windows, the operating system will assign a COM-port. A serial port is typically named COM<x> where <x> is a number between 1 and 256. You can observe the COM-port in the device manager. When the devices are displayed by type, the COM-port appears under the "Ports (COM & LPT)" branch. You can select to view devices by connection which renders a tree structure.

Thermometer Software

The thermometer software is an Arduino sketch. **Arduino IDE v1.8.5** was used for developing and testing the software. The source code is made up of .ino files. You will hopefully find many useful remarks and comment blocks in these source files to aid in understanding the code. There are two ways of uploading the thermometer software to the Arduino Nano:

- Use avrdude to upload the binary file.
- Upload from within the Arduino IDE. *This is the most commonly used method.*

Either way you need to install the Arduino IDE.[5]

To upload the software with avrdude, first put file **in_9_thermometer.ino.hex** (located in the .zip archive of binary files) in a directory, and then invoke the following command from the command prompt in that directory:

```
bin> path-to-avr\bin\avrdude -C path-to-avr\etc\avrdude.conf -v -p atmega328p -c arduino
-PCOM5 -b57600 -D -Uflash:w:in_9_thermometer.ino.hex:i
```

Replace **`bin`** with the path to the directory that contains the .hex file. Replace **`path-to-avr`** with the path to the tools directory in the Arduino IDE directory structure. For example:

C:\arduino-1.8.5\hardware\tools\avr

avrdude will output information like this:

```
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:
\arduino-1.8.5\hardware\tools\avr\etc\avrdude.conf"

         Using Port                    : COM5
         Using Programmer              : arduino
         Overriding Baud Rate          : 57600
         AVR Part                      : ATmega328P
         Chip Erase delay              : 9000 us
         PAGEL                         : PD7
         BS2                           : PC2
         RESET disposition             : dedicated
         RETRY pulse                   : SCK
         serial program mode           : yes
         parallel program mode         : yes
         Timeout                       : 200
         StabDelay                     : 100
         CmdexeDelay                   : 25
         SyncLoops                     : 32
         ByteDelay                     : 0
         PollIndex                     : 3
         PollValue                     : 0x53
         Memory Detail                 :

                                  Block Poll               Page                       Polled
           Memory Type Mode Delay Size  Indx Paged  Size   Size #Pages MinW  MaxW   ReadBack
           ----------- ---- ----- ----- ---- ------ ------ ---- ------ ----- ----- ---------
           eeprom        65    20     4    0 no       1024    4      0  3600  3600 0xff 0xff
           flash         65     6   128    0 yes     32768  128    256  4500  4500 0xff 0xff
           lfuse          0     0     0    0 no          1    0      0  4500  4500 0x00 0x00
           hfuse          0     0     0    0 no          1    0      0  4500  4500 0x00 0x00
           efuse          0     0     0    0 no          1    0      0  4500  4500 0x00 0x00
           lock           0     0     0    0 no          1    0      0  4500  4500 0x00 0x00
           calibration    0     0     0    0 no          1    0      0     0     0 0x00 0x00
           signature      0     0     0    0 no          3    0      0     0     0 0x00 0x00

         Programmer Type : Arduino
         Description     : Arduino
         Hardware Version: 2
```

[5] https://www.arduino.cc/en/software

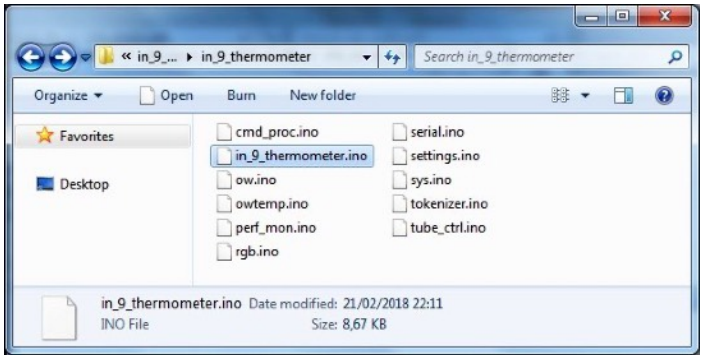
```
         Firmware Version: 1.16
         Vtarget         : 0.0 V
         Varef           : 0.0 V
         Oscillator      : Off
         SCK period      : 0.1 us

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% -0.00s

avrdude: Device signature = 0x1e950f (probably m328p)
avrdude: safemode: hfuse reads as 0
avrdude: safemode: efuse reads as 0
avrdude: reading input file "in_9_thermometer.ino.hex"
avrdude: writing flash (15152 bytes):

Writing | ################################################## | 100% 4.78s

avrdude: 15152 bytes of flash written
avrdude: verifying flash memory against in_9_thermometer.ino.hex:
avrdude: load data flash data from input file in_9_thermometer.ino.hex:
avrdude: input file in_9_thermometer.ino.hex contains 15152 bytes
avrdude: reading on-chip flash data:

Reading | ################################################## | 100% 3.72s

avrdude: verifying ...
avrdude: 15152 bytes of flash verified

avrdude: safemode: hfuse reads as 0
avrdude: safemode: efuse reads as 0
avrdude: safemode: Fuses OK (E:00, H:00, L:00)

avrdude done.  Thank you.
```

After this, the thermometer is ready to be set up as described later.

However, the most common way to upload the thermometer software is from within the Arduino IDE.

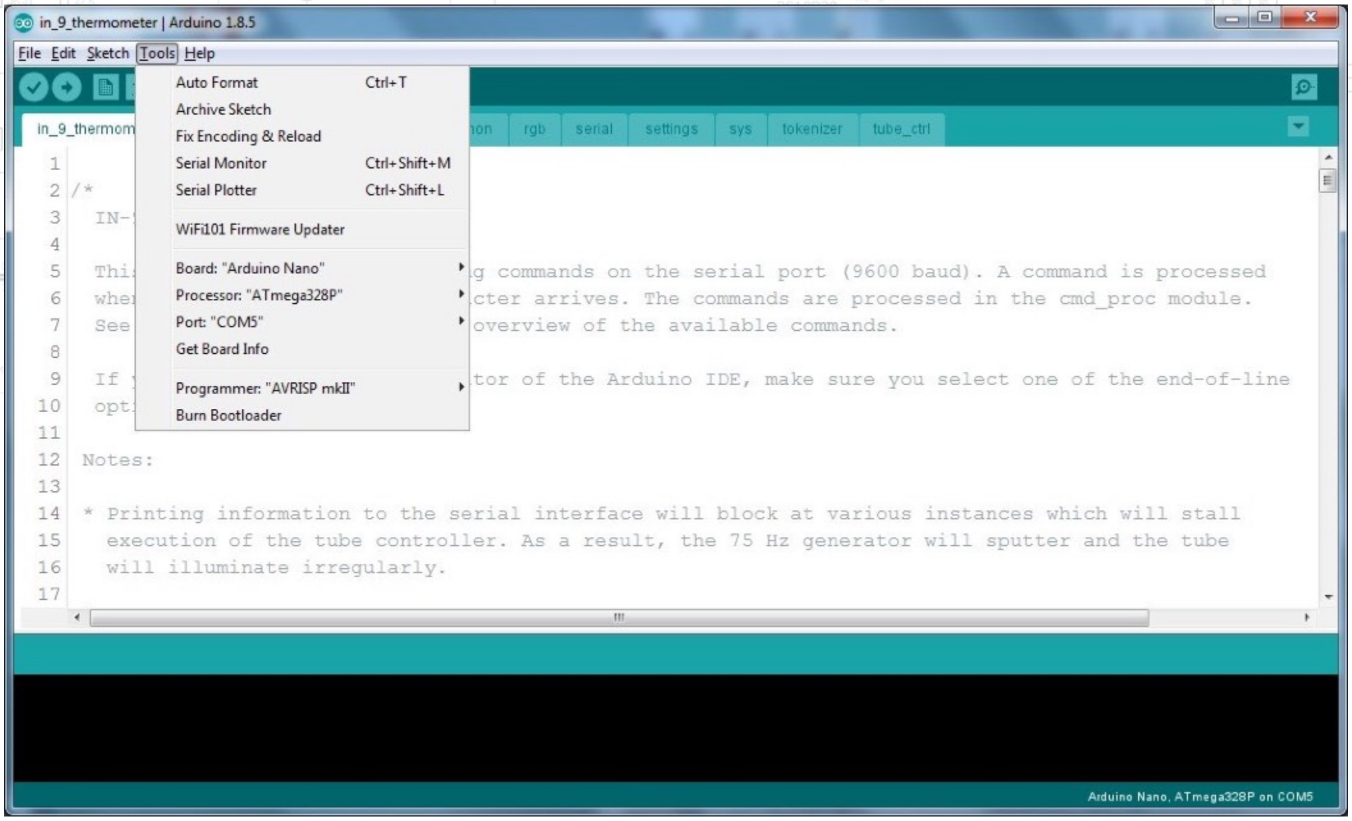
The source code consists of multiple .ino source files. The main file is called **in_9_thermometer.ino**, it has the same name (without .ino) as the containing folder.

Open **in_9_thermometer.ino** in the Arduino IDE to open up the Sketch. The IDE shows all files in tabs, starting with the main file and followed by the other files in the folder sorted alphabetically.



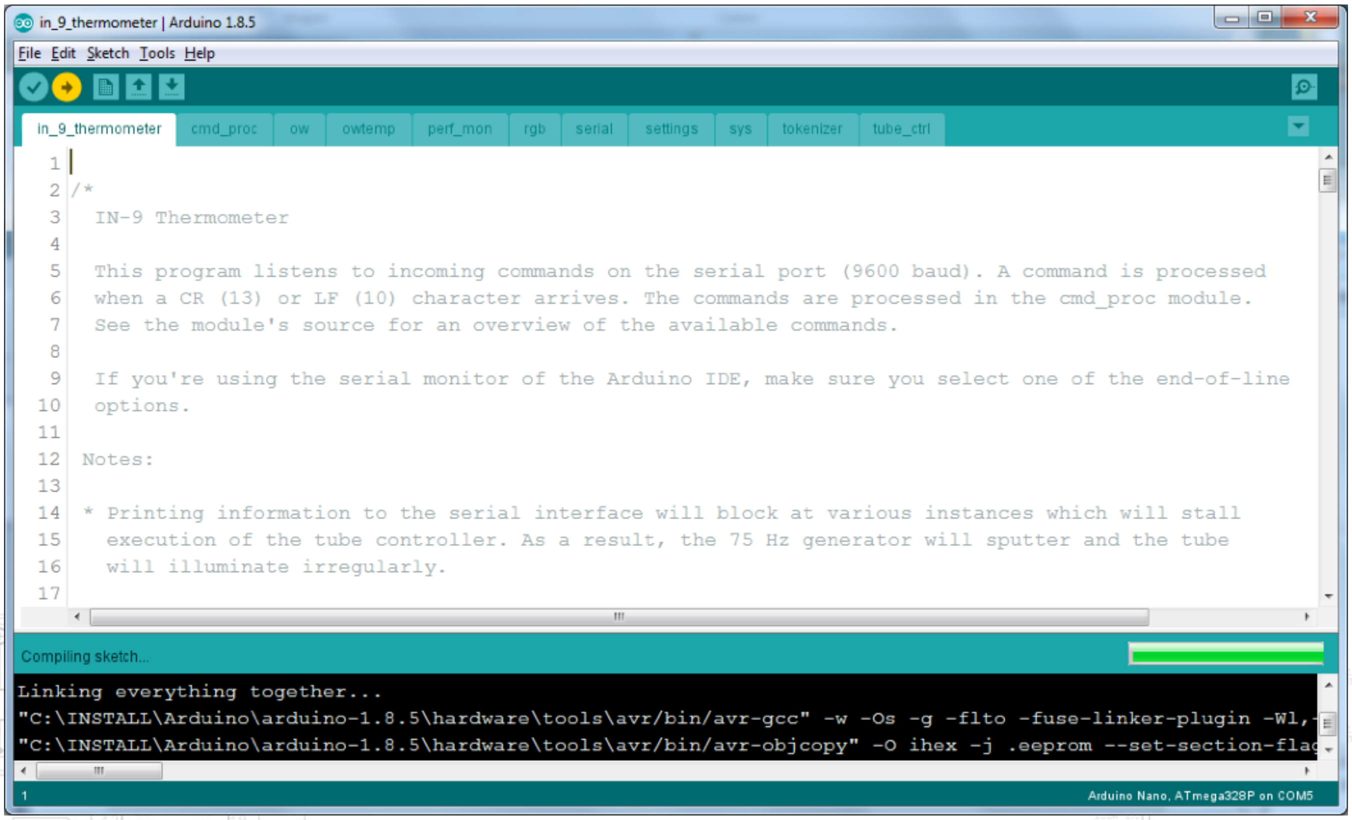
cmd_proc.ino
in_9_thermometer.ino
ow.ino
owtemp.ino
perf_mon.ino
rgb.ino
serial.ino
settings.ino
sys.ino
tokenizer.ino
tube_ctrl.ino

in_9_thermometer.ino Date modified: 21/02/2018 22:11
INO File Size: 8,67 KB


in_9_thermometer | 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 Nano"
Processor: "ATmega328P"
Port: "COM5"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
in_9_thermom
rgb serial settings sys tokenizer tube_ctrl
Arduino Nano, ATmega328P on COM5

```

/*
  IN-

  Thi                                  g commands on the serial port (9600 baud). A command is processed
  whe                                  cter arrives. The commands are processed in the cmd_proc module.
  See                                  overview of the available commands.

  If                                   tor of the Arduino IDE, make sure you select one of the end-of-line
  opt

Notes:

 * Printing information to the serial interface will block at various instances which will stall
   execution of the tube controller. As a result, the 75 Hz generator will sputter and the tube
   will illuminate irregularly.

```

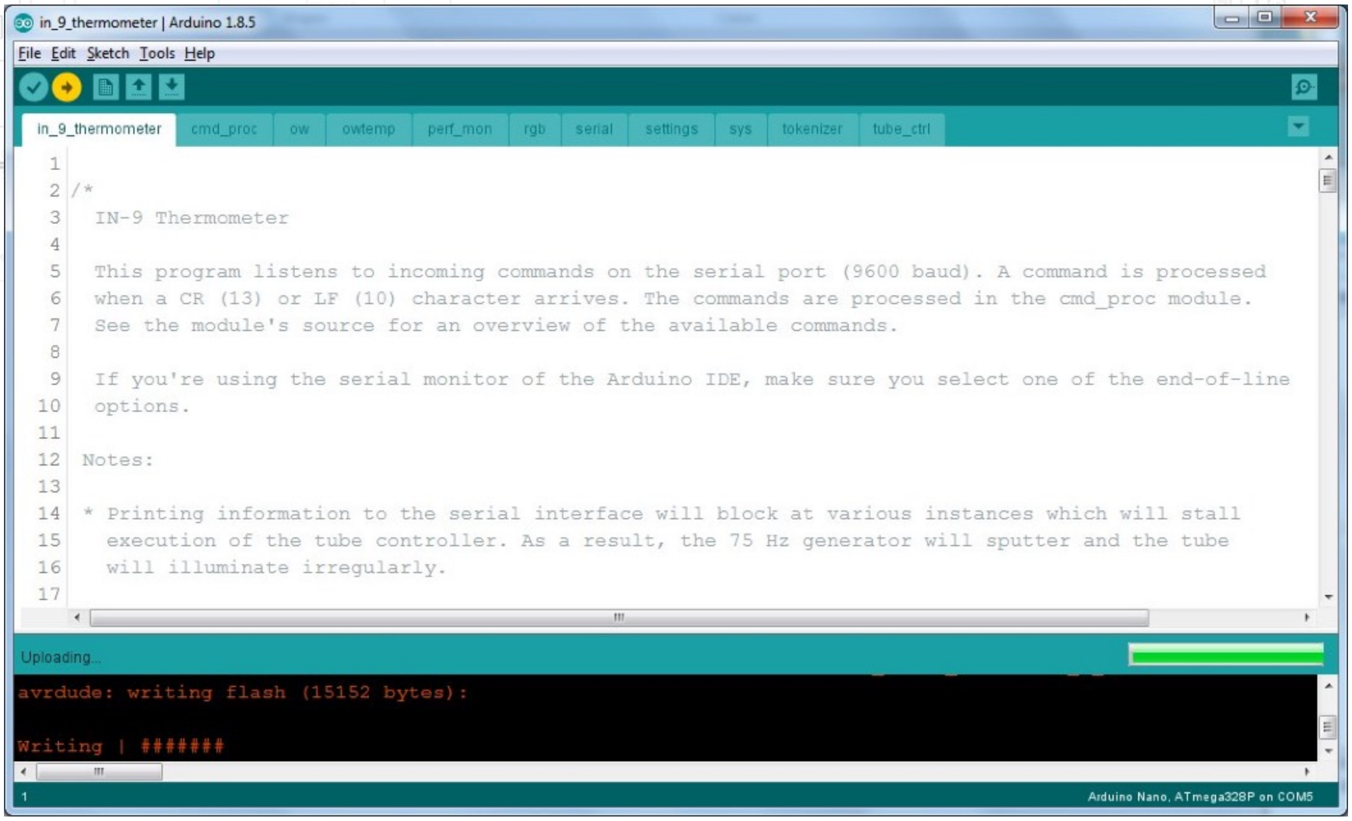
Before you start building the sources, you have to set up the IDE for use with the Arduino Nano. In the Tools menu, select board type “Arduino Nano” and select the COM-port assigned to your thermometer.


in_9_thermometer | Arduino 1.8.5
File Edit Sketch Tools Help
in_9_thermometer cmd_proc ow owtemp perf_mon rgb serial settings sys tokenizer tube_ctrl
Compiling sketch...
1
Arduino Nano, ATmega328P on COM5

```

/*
  IN-9 Thermometer

  This program listens to incoming commands on the serial port (9600 baud). A command is processed
  when a CR (13) or LF (10) character arrives. The commands are processed in the cmd_proc module.
  See the module's source for an overview of the available commands.

  If you're using the serial monitor of the Arduino IDE, make sure you select one of the end-of-line
  options.

Notes:

 * Printing information to the serial interface will block at various instances which will stall
   execution of the tube controller. As a result, the 75 Hz generator will sputter and the tube
   will illuminate irregularly.

```

```
Linking everything together...
"C:\INSTALL\Arduino\arduino-1.8.5\hardware\tools\avr/bin/avr-gcc" -w -Os -g -flto -fuse-linker-plugin -Wl,
"C:\INSTALL\Arduino\arduino-1.8.5\hardware\tools\avr/bin/avr-objcopy" -O ihex -j .eeprom --set-section-flag
```

Now you can build and upload the software. In the Arduino IDE, this process is simply called “Upload”. Select menu Sketch -> Upload, or press CTRL-U to proceed.


in_9_thermometer | Arduino 1.8.5
File Edit Sketch Tools Help
in_9_thermometer
cmd_proc
ow
owtemp
perf_mon
rgb
serial
settings
sys
tokenizer
tube_ctrl
Uploading...
Arduino Nano, ATmega328P on COM5

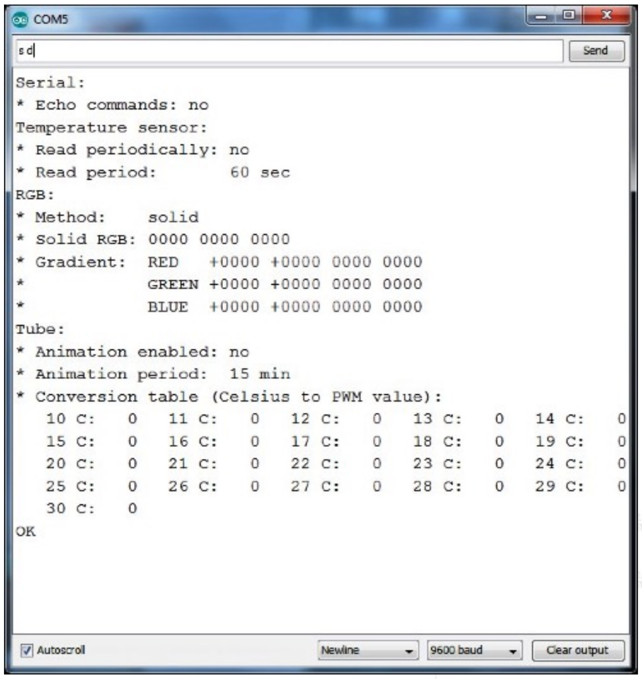
```
/*
 IN-9 Thermometer

 This program listens to incoming commands on the serial port (9600 baud). A command is processed
 when a CR (13) or LF (10) character arrives. The commands are processed in the cmd_proc module.
 See the module's source for an overview of the available commands.

 If you're using the serial monitor of the Arduino IDE, make sure you select one of the end-of-line
 options.

Notes:

* Printing information to the serial interface will block at various instances which will stall
  execution of the tube controller. As a result, the 75 Hz generator will sputter and the tube
  will illuminate irregularly.

```

```
avrdude: writing flash (15152 bytes):

Writing | #######
```

After the IDE has successfully uploaded the binary file (.hex) to the Arduino Nano, the thermometer is ready to communicate over the serial interface. You can use the IDE's serial monitor for this purpose. Select menu Tools -> Serial Monitor, or press CTRL-SHIFT-M to open up the serial monitor. Set the baudrate to 9600 and select something different from "no line ending".

IMPORTANT! Each time you open the serial interface, the board will reset. This behavior is specified by Arduino and occurs with all Arduino-compatible boards, independent of operating system and serial monitor program (e.g. Arduino IDE's serial monitor, PuTTY, netcat, ...). Therefore, beware, when you change settings, then close and reopen the serial interface before writing the settings to non-volatile memory (a.k.a. EEPROM), the changes will be lost.

You can send commands to the thermometer from the serial monitor. For example, command **s d** (a short version of **settings dump**) will report a list of all current settings.



```
s d
```

```
Serial:
* Echo commands: no
Temperature sensor:
* Read periodically: no
* Read period:       60 sec
RGB:
* Method:    solid
* Solid RGB: 0000 0000 0000
* Gradient:  RED   +0000 +0000 0000 0000
*            GREEN +0000 +0000 0000 0000
*            BLUE  +0000 +0000 0000 0000
Tube:
* Animation enabled: no
* Animation period:  15 min
* Conversion table (Celsius to PWM value):
   10 C:    0    11 C:    0    12 C:    0    13 C:    0    14 C:    0
   15 C:    0    16 C:    0    17 C:    0    18 C:    0    19 C:    0
   20 C:    0    21 C:    0    22 C:    0    23 C:    0    24 C:    0
   25 C:    0    26 C:    0    27 C:    0    28 C:    0    29 C:    0
   30 C:    0
OK
```

At this point, no settings are set, unless your Arduino Uno was already set up for the thermometer. To fill in the settings, you have to send a series of commands. You can manually type each command in the serial monitor. Alternatively, you can put the commands in a text file and send its contents to the serial port. Since the serial monitor in the Arduino IDE isn't capable of sending a file or pasting multiple lines from the clipboard, it is best to rely on another serial monitor. A good choice is PuTTY[6] in Windows and Linux. Another good choice is netcat in Linux.

Commands that change a setting have a 2nd keyword called set followed by the actual setting e.g. **tube set animperiod 5**. Settings are always changed in RAM. Command **s d** (short for **settings dump**) reports all current settings in RAM.

To make the settings permanent, issue command **s w** (short for **settings write**). This command writes the settings in RAM to non-volatile memory.

Here are the contents of an example text file with setting commands. You can copy the text from this document in your own text editor (notepad etc.) and adjust the settings to your liking.

```
settings clear

ts set read yes
ts set readperiod 60

rgb set solid id 14
rgb set grad id 1
rgb set method grad
rgb set grad use

tube set anim yes
tube set animperiod 5

tube value 3
tube set temp 10

tube value 39
tube set temp 15

tube value 76
tube set temp 20

tube value 114
tube set temp 25

tube value 160
tube set temp 30

settings write
```

Note that the set of **tube value** and **tube set temp** commands is specific for each IN-9 tube. The example only contains five settings. Normally you would set all 21 values in the conversion table.

[6] https://www.putty.org

Setting up the Conversion Table

The thermometer software relies on a conversion table to translate the measured temperature in Celsius to a PWM value. The software uses the PWM value to drive the IN-9 tube to the height that indicates the measured temperature on the acrylic scale. Since each tube is slightly different, it is best to set up the conversion table for your individual thermometer.

Send commands **ts set r n** (short for **ts set read no**) and **t set a n** (short for **tube set anim no**) to turn off periodic reading of temperature and tube animation respectively. This way the thermometer function won't drive the tube when you're setting up the conversion table.

The conversion table has 21 entries, one for each temperature ranging from 10°C to 30°C. You can dump the conversion table with command **s d** (short for **settings dump**). Initially, all entries are zeroes. Zero means there is no PWM value associated with the corresponding temperature. A value between 1 to 255 indicates a PWM value for the entry's corresponding temperature. PWM values typically range from 1 to 160.

Each time the thermometer measures the temperature, it looks up the corresponding PWM value in the table. If a non-zero value is present, this value is used for driving the tube. However, if the entry in the table is zero, there's no PWM value to drive the tube. Instead the software scans adjacent entries and when it finds both an upper entry and a lower entry that are non-zero, it linearly interpolates the values to produce a PWM value for the measured temperature.

Let's give an example. The entry for 20°C is 76, 25°C has 114, and 21°C..24°C all have a value of zero. The temperature sensor measures 23°C. Since there's no PWM value for this temperature, the software looks for adjacent entries that are non-zero, being 20°C and 25°C. The software interpolates between the corresponding PWM values 76 and 114, the result is 98. You can actually recreate this example as follows:

1. Send command **ts f 23**. This will force-feed 23°C, perform a conversion to PWM value, and drive the tube.
2. Dump the result with command **t d** (short for **tube dump**). It'll show PWM value 98.

Note that the thermometer shows the temperature with a 0.5°C precision. When the measured temperature ends in .5, there is no entry in the table, and the software interpolates between PWM values.

To set up the conversion table, you have to manually set PWM values to drive the tube to a height that corresponds with a temperature in Celsius on the scale. When you've found a suitable PWM value for the temperature, you tell the software to put the PWM value in the table.

For instance, say you want to determine the PWM value for 25°C. First set a PWM value like 110 with command **t v 110** (short for **tube value 110**). Now observe the tube. It's best to look straight at the tube rather than from above. If the height corresponds with 25°C on the scale then you're done! If not, you have to try a lower or higher PWM value, depending on whether the tube is driven too high or too low. Once you've found the correct PWM value, you can write it to the table entry for 25°C with command **t set t 25** (short for **tube set temp 25**).

At any time you can view the contents of the conversion table. The output of command **s d** contains the conversion table, for example:

```
* Conversion table (Celsius to PWM value):
  10 C:   3   11 C:   0   12 C:   0   13 C:   0   14 C:   0
  15 C:  39   16 C:   0   17 C:   0   18 C:   0   19 C:   0
  20 C:  76   21 C:   0   22 C:   0   23 C:   0   24 C:   0
  25 C: 114   26 C:   0   27 C:   0   28 C:   0   29 C:   0
  30 C: 160
```

The more PWM values you put in the table, the more precise the thermometer will render the temperature on the acrylic scale. Don't forget to write the settings to non-volatile memory with the **s w** command. It may take some time to set up the conversion table, hence it is recommended to send the command **s w** regularly.

When you're finished with the conversion table, send commands **ts set r y** (short for **ts set read yes**) and **t set a y** (short for **tube set anim yes**) to turn back on periodic reading of temperature and tube animation. Then send command **s w** to make the settings persistent.

Driving the IN-9 Tube

You can manually drive the tube with command **t v** (short for **tube value**), e.g. command **t v 125** drives the tube with a PWM value of 125.

Tube Animation

It's important that the tube is driven to its maximum current every so often, else the tube may degrade over time and become faulty. The software incorporates a so-called tube animation that drives the tube in a specific pattern at a specified interval. Send command tube **set anim yes** to enable the animation and command **tube set animperiod** to set the animation interval in minutes.

Troubleshooting - Transistor Issue

If the thermometer remains at full scale display without the possibility to control the tube via software, one of the MPSA42 transistors (T2, T3) is probably defective and conducts without base current applied. This seems to happen occasionally with this type of transistors due to production faults. Measure the voltage across R3 and R4. The resistor with the highest voltage is connected to the faulty transistor. Connect all the three legs of the transistor together with a blob of solder. Heat it again until it's melted and remove the transistor by pulling on it from the component side of the PCB. Clean up the solder pads with solder wick and use a bit of rubbing alcohol to clean flux stains. Then replace the transistor with a new one.

Commands

The thermometer software accepts an extensive list of commands. There are several groups of commands, each starting with a specific keyword e.g. **tube**. Many commands can be abbreviated. For example, **t set ap 5** is short for **tube set animperiod 5**.

Commands that change a setting have a 2nd keyword called set followed by the actual setting e.g. **tube set anim yes**. Settings are always changed in RAM. To make the settings permanent, issue command **settings write**.

Note that the **settings** command group doesn't change actual settings. These commands manage the settings as a whole.

The commands are documented and implemented in the source file **cmd_proc.ino**.

Here's a list of recurring parameters. Parameters that are specific to a command are explained with the command.

```
Recurring parameters:

  RED      0..4095
  GREEN    0..4095
  BLUE     0..4095
  YN       yes|y,no|n
```

Here's a list of all possible commands.

```
serial|ser                    Serial interface
∟ set                         Settings
  ∟ echocmd|ec YN             Enable or disable echoing of commands on the command
                                interface
```

```
sys                           System
∟ reset                       System reset
```

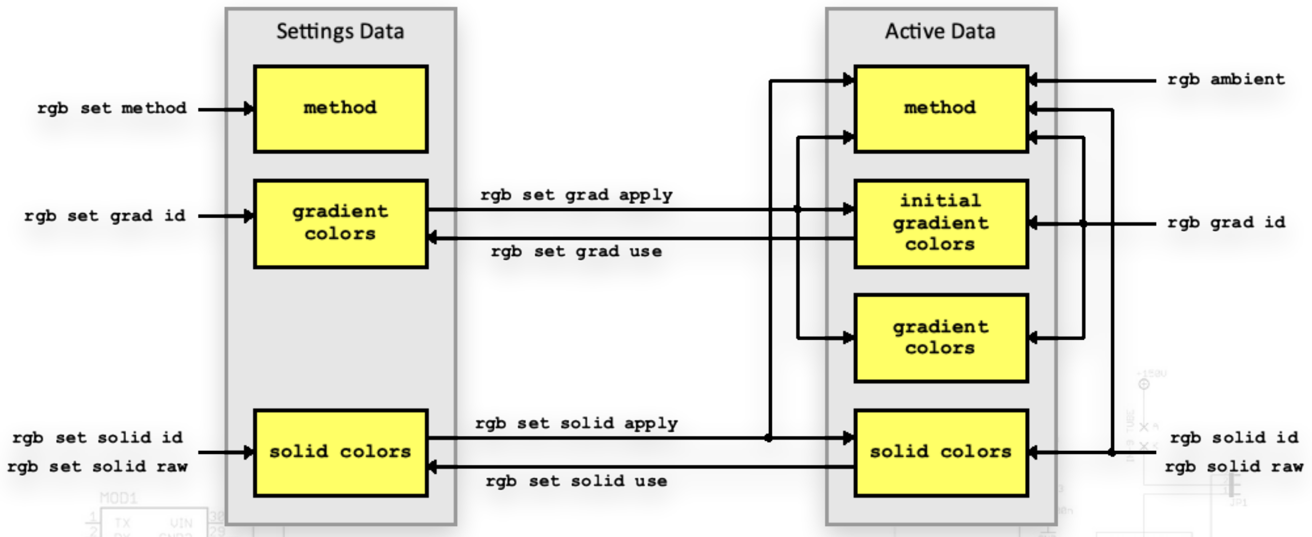
```
ts                            Temperature sensor
∟ read|r                      Read temperature sensor now
∟ force|f [+|-] N [F]         Force-feed a temperature (N=0..32767, F=0..4)
∟ dump|d                      Dump information
∟ set                         Settings
  ∟ read|r YN                 Enable or disable periodic reading
  ∟ readperiod|rp S           Specify interval period for periodic reading, seconds
                                (S=1..60)
```

```
tube|t                        Tube controller
∟ value|v N                   Set tube PWM value (N=0..255)
∟ anim|a                      Start tube animation now
∟ dump|d                      Dump information
∟ set                         Settings
  ∟ anim|a YN                 Enable or disable periodic animation
  ∟ animperiod|ap MIN         Specify interval period for periodic animation, minutes
                                (MIN=1..15)
  ∟ temp|t N                  Assign current PWM value to the specified temperature in
                                Celsius (N=10..30)
```

```
rgb                                RGB driver
└ solid|s                          Select solid RGB colors
  └ id N                           Select scheme from list (N=0..17)
  └ raw RED GREEN BLUE             Specify RGB values
└ grad|s                           Select animated gradient colors
  └ id N                           Select scheme from list (N=0..5)
└ ambient|a                        Select ambient colors
└ set                              Settings
  └ method|m                       Color method
    └ solid|s                      Use solid RGB colors
    └ grad|g                       Use animated gradient colors
    └ ambient|a                    Use ambient
  └ solid|s                        Solid RGB colors
    └ id N                         Select scheme from list (N=0..17)
    └ raw RED GREEN BLUE           Specify RGB values
    └ use                          Copy active solid RGB colors to settings
    └ apply                        Apply solid RGB colors in settings as active
  └ grad|g                         Animated gradient colors
    └ id N                         Select scheme from list (N=0..5)
    └ use                          Copy active gradient colors to settings
    └ apply                        Apply gradient colors in settings as active
```

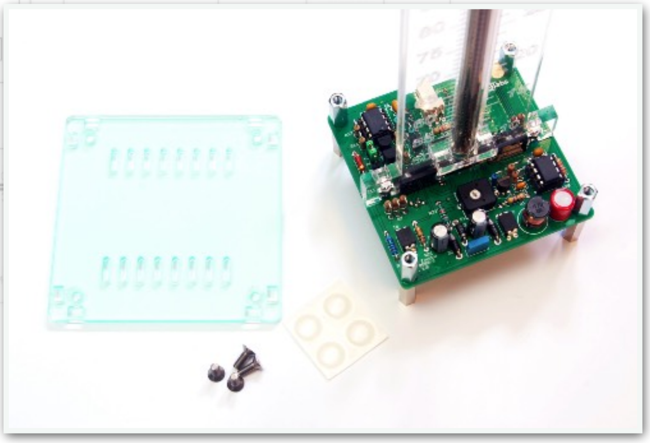
```
settings|s                         Settings
└ write|w                          Write settings from RAM to EEPROM
└ read|r                           Read settings from EEPROM to RAM, verify, apply
                                     default settings if invalid
└ reset                            Reset settings in RAM
└ clear                            Clear settings in EEPROM
└ dump|d                           Dump information
```

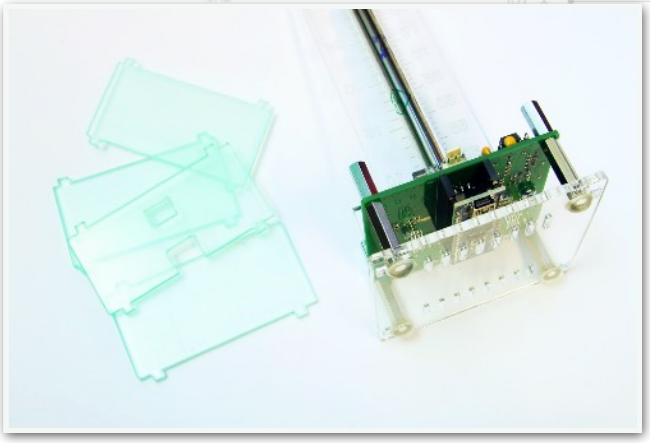
The number of **rgb** commands is extensive. This is because the software manages two sets of RGB data, one for settings and one for activity (the colors actually being shown). An early version of the software had one data set, but this design led to confusion when one was updating settings while fiddling with the RGB colors. Therefore it was decided to split up the RGB data into two data sets. The diagram below shows how the various rgb commands influence the data sets. Each command induces a data flow through the associated arrow(s).

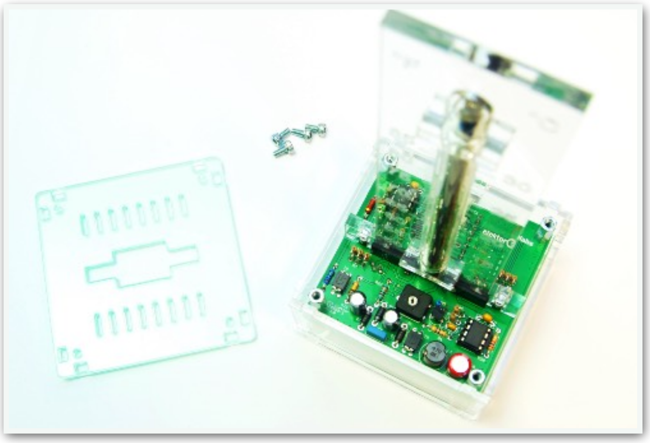


The active set stores a copy of the initial gradient colors since the colors and thus the data values are constantly changing when the animated gradient color method is active.

Assembling the Enclosure





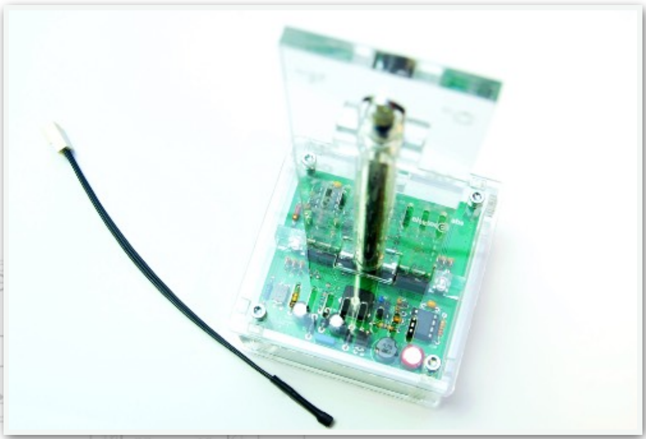


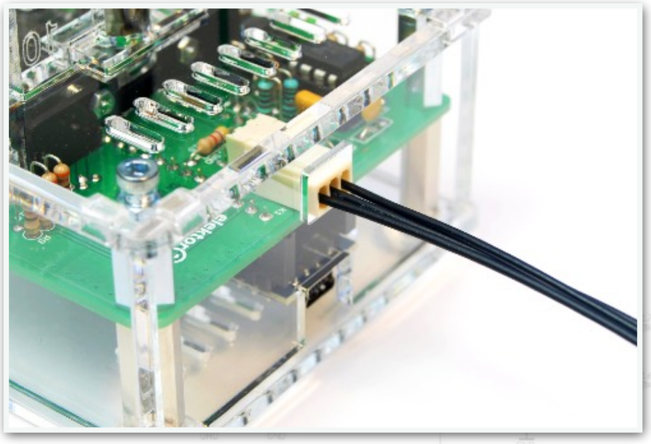
Remove the protective coatings from the bottom panel and attach it to the 20 mm standoffs mounted to the PCB, using four countersunk M3x8 machine screws.

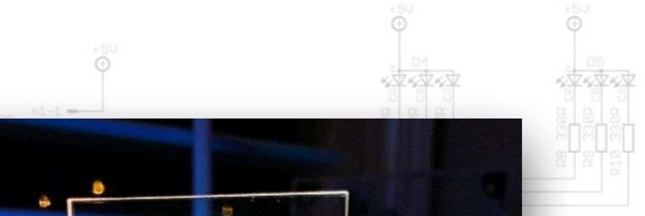
Stick four self-adhesive rubber feet on top of the screw heads.

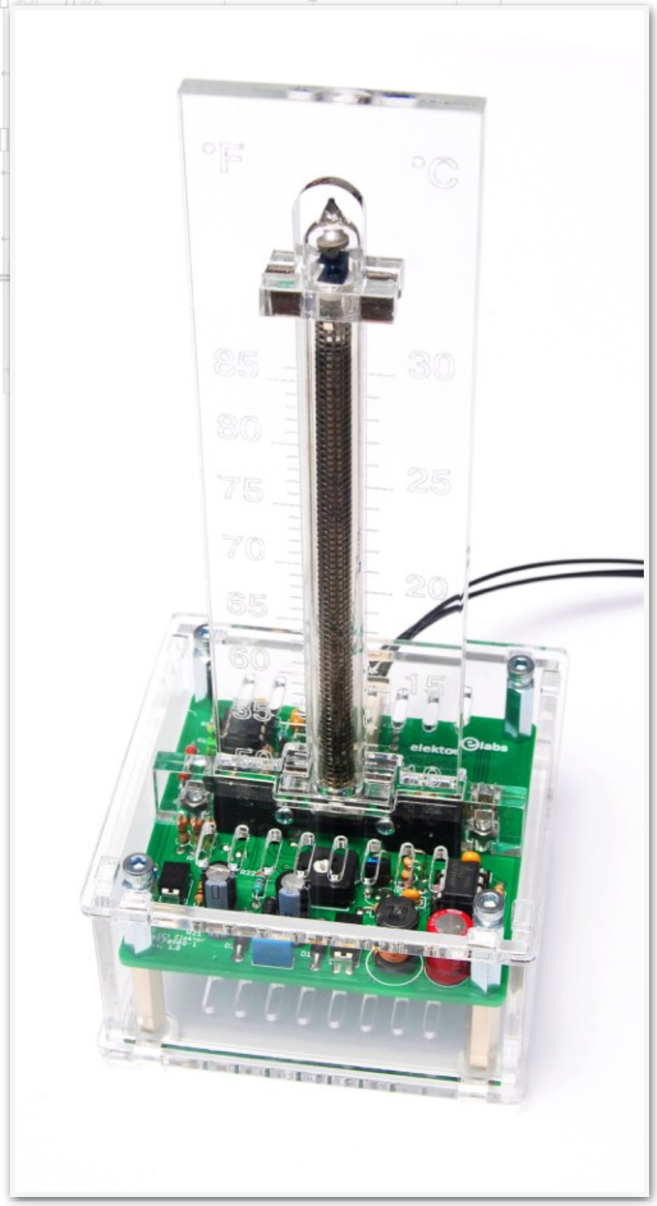
Remove the protective coatings from the front, back and side panels and put them in place.

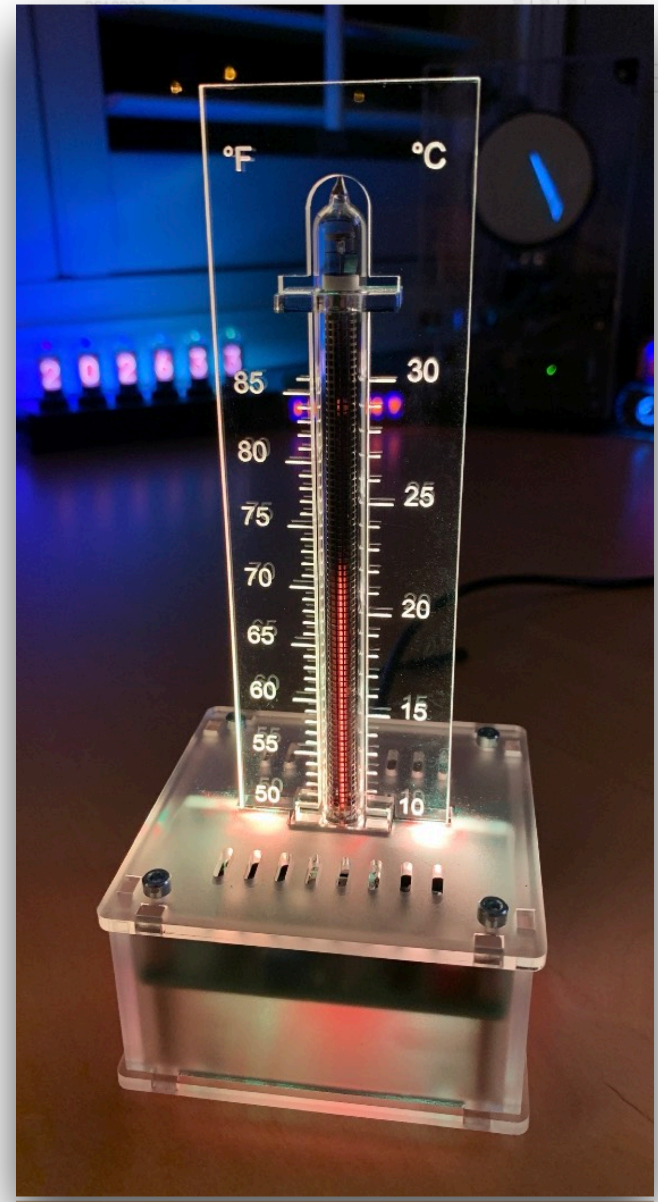
Finally remove the protective coatings of the top panel. Slide the top panel over the scale down until it rests on the 14 mm standoffs. Secure the panel using four M3x6 screws with hexagonal head. Connect the temperature sensor to K1 on the PCB of the thermometer. Remove any fingerprints, especially on the scale, using a piece of cotton cloth. If a dry cloth does not work well enough, don't use any solvents including (rubbing) alcohol as this may cause tension cracks. Instead use a dampened cloth with water and kitchen soap instead. As an alternative, you can also use cleaning cloths for glasses or a specialty plastic cleaner suitable for acrylic.











You've completed your IN9 Thermometer. Enjoy!