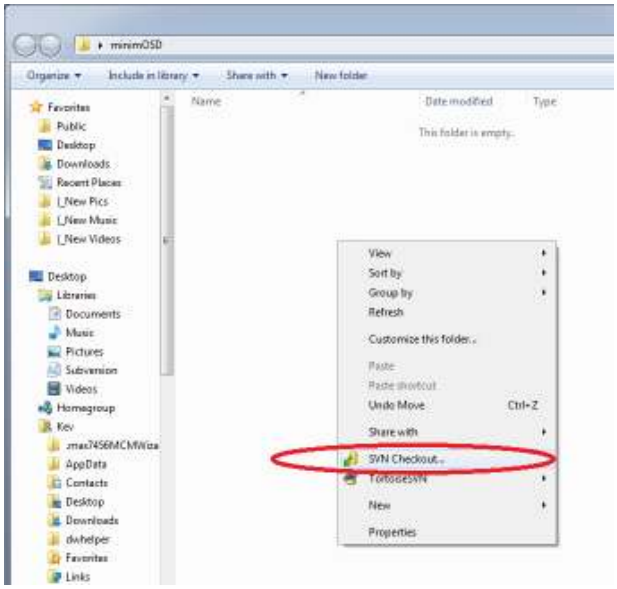
Beginner's Guide – Howto Modify the MimimOSD-Extra Code

This guide will attempt to teach those with no programming experience the basic concepts required to make minor cosmetic modifications to the minimOSD firmware. As with all things open source, this guide builds upon works of others. Credit is due to Gabor and Iskess, and the many online community members who share their ideas freely. Without folks like these, I wouldn't know Jack from Jill. Anyhow, let's get right down to business…

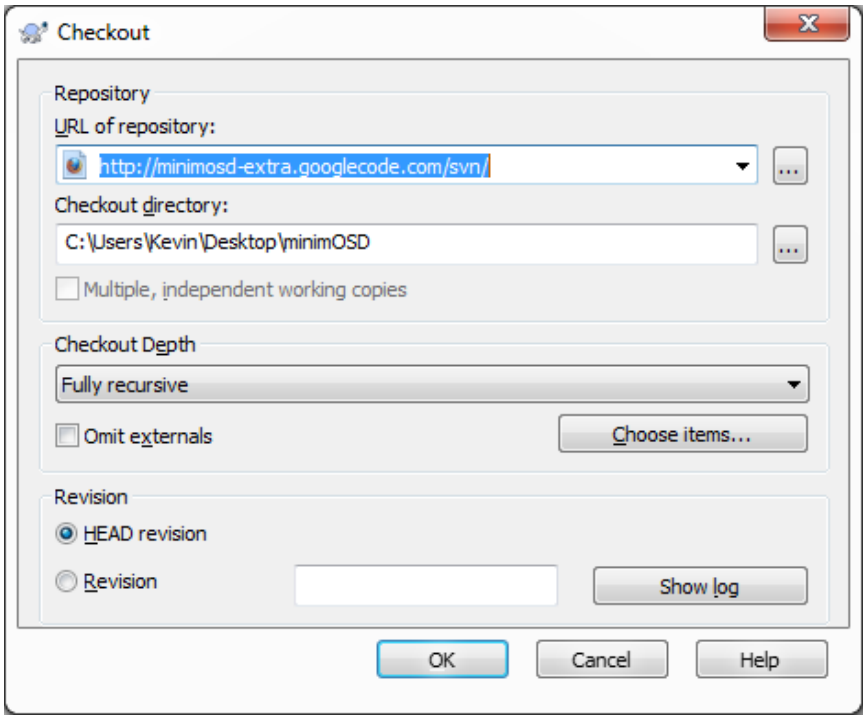
Download & Compile the source in Arduino:

First, download and install Tortoise SVN client software on your computer:
http://tortoisesvn.net/downloads.html

Go to the folder where you want to keep your minimOSD-extra source code, rightclick and select "SVN Checkout" from the context menu as shown below:



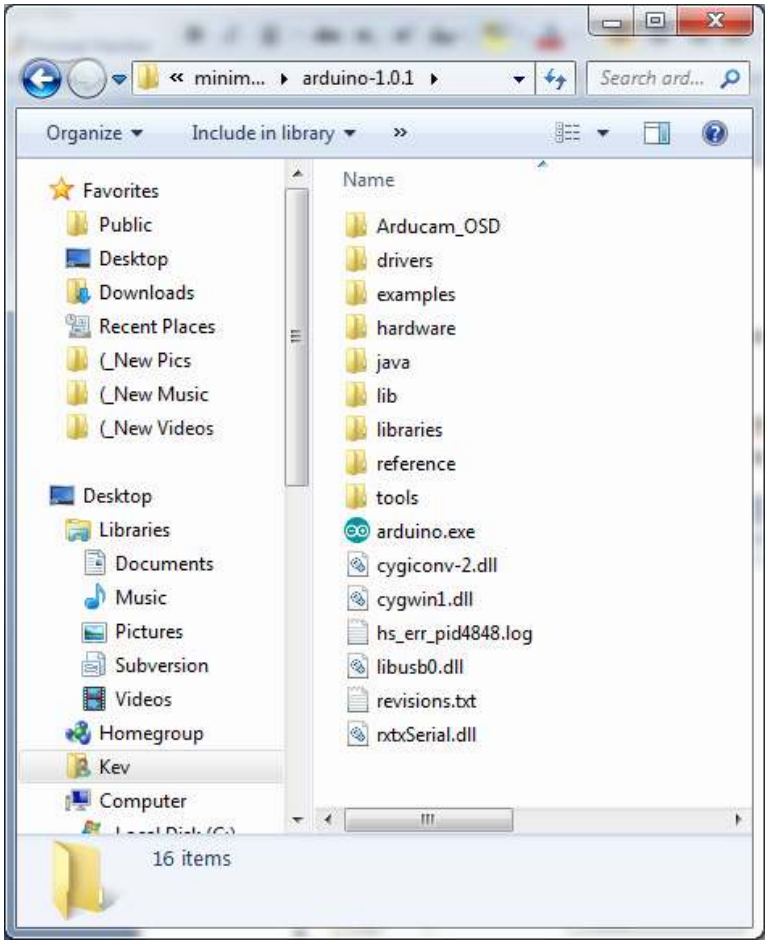
Enter the minimOSD-extra SVN checkout address and download the head revision as shown below:



Click OK, and wait for the ~15MB download to finish. You should now have 2 folders named “trunk” and “wiki”. All of the source code we need to compile firmware is in “trunk”. Now let’s download the Arduino IDE, version 1.0.1:

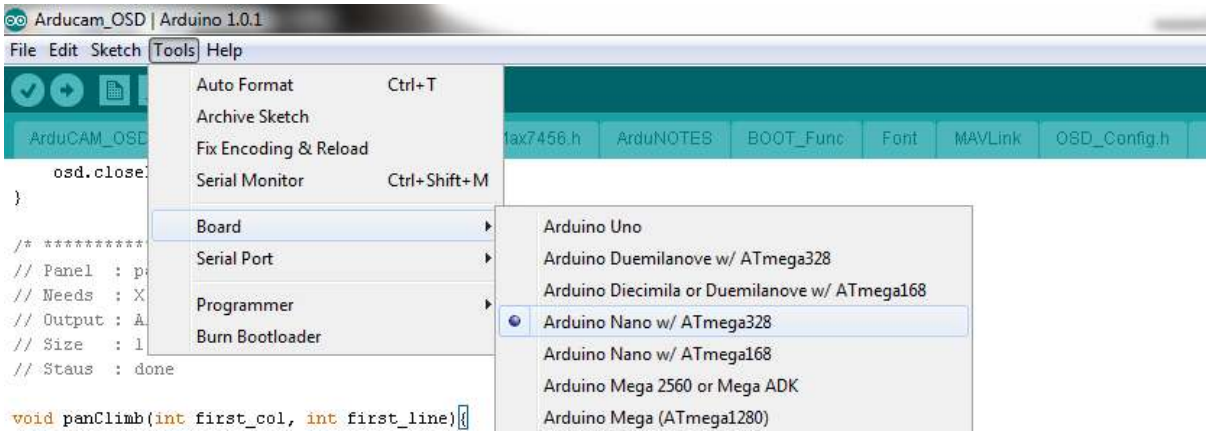
http://arduino.cc/en/Main/Software

Unzip the contents of this zip file in to your folder of choice. Now go back to the folder named “trunk”, and copy the 2 folders named “MinimOsd-Extra_Pre release_Beta” and “libraries” in to the Arduino 1.0.1 folder. Answer yes if asked to merge “libraries” folders; Arduino comes with some libraries, but we need to make sure Arduino also has the minimOSD libraries as well. Now rename the “MinimOsd-Extra_Pre release_Beta” folder (you just copied) to “Arducam_OSD”. Organization of your sketchbook directories is important. When you are done, your Arduino folder should look like this:



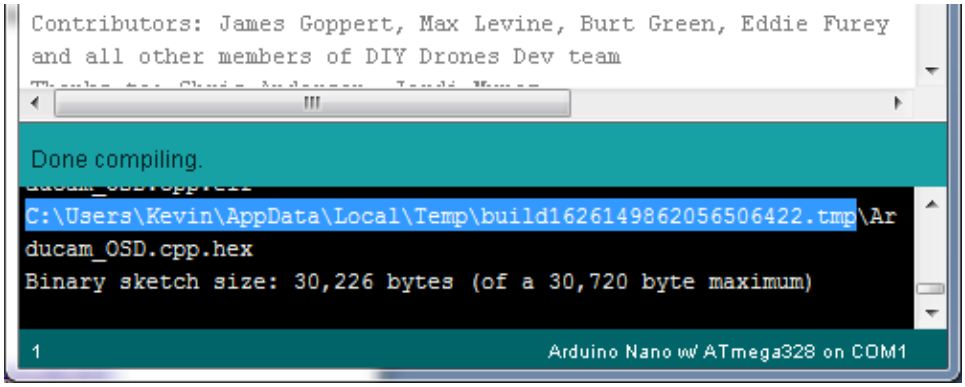
Now double click “arduino.exe”, and when Arduino opens click “File\Preferences”, and under “Sketchbook location:” browse to your “arduino-1.0.1” directory, and select the verbose “compilation” checkbox as shown below, then click OK:

Exit and restart Arduino, and click “File\Sketchbook”, and you should see “Arducam_OSD” at the top of the menu. If not you got something wrong, and must go back to fix it or compilation will not work. If you see it, click on it to open up the minimOSD-extra sketchbook in a new window (you can close the blank window now). Now select the Nano 328 board under “Tools\Board\Arduino Nano w/ Atmega328” as shown below:



Now test to see if the base code you downloaded compiles correctly. Click “Verify”… the round button with a check mark. Wait a second for your computer to compile the code; you will see some red colored errors go by on the verbose output below which you can ignore. If compilation was successful you will

see “Done compiling” as shown below, and just below that is the directory where the hex file is temporarily located:



Copy that directory location (shown highlighted above) to your clipboard (ctrl+V), and paste it in to Windows Explorer to open the temp folder with your fresh new hex file inside. If you close Arduino now, that folder will get deleted. So be sure to copy that hex file to a safe place before exiting, and remember this every time you play in Arduino. Yeah, now you’ve successfully compiled your own hex file for minimOSD. Take this time to enjoy your cold beverage of choice, and if you haven’t already, scroll around and click the tabs at the top of the sketchbook to skim over the beautifully complex source code. Each tab represent a file used in the code.

Editing the “OSD_Panels“ file:

Now that we know how to compile with the Arduino IDE, we must get our heading straight with the minimOSD source. The more C programming you know, the more you will be able to do with the code. This guide focuses on those who have never seen code before, and think this project would make a good start towards a life of scripting. ;)

The file we mostly interested in editing is named “OSD_Panels”; click that tab, and get familiar with its layout. Each panel is broken down in to a section, most of which are only 6 or so lines, some longer. This guide will focus more on the smaller, simpler to edit panels. To make it really simple, most of the popular modifications to minimOSD only require editing one line per panel. We’ll modify the climb rate panel for our example. Here is the snippet of code from the head source at the time of writing this guide, with the important line highlighted:

```
/* ********************************************************************* */
// Panel  : panClimb
// Needs  : X, Y locations
// Output : Alt symbol and altitude value in meters from MAVLink
// Size   : 1 x 7Hea  (rows x chars)
// Staus  : done

void panClimb(int first_col, int first_line){
   osd.setPanel(first_col, first_line);
```

```
    osd.openPanel();
    osd.printf("%c%3.0f%c",0x16, (osd_climb), 0x88);
    osd.closePanel();
}
```

That highlighted line starts with “%c%3.0f%c”. This tells the print function to print a simple character (%c), then a 3 digit precision float with 0 digits after the decimal (%3.0f), then another simple character. These simple characters are the ones that we edit with the Max7456 Character Editor:
http://minimosd-extra.googlecode.com/svn/trunk/Tools/MAX7456Charwizard.jar

Here is where I learned about this tool, which makes it easy to clean up minimOSD:
http://www.rcgroups.com/forums/showthread.php?t=1830845

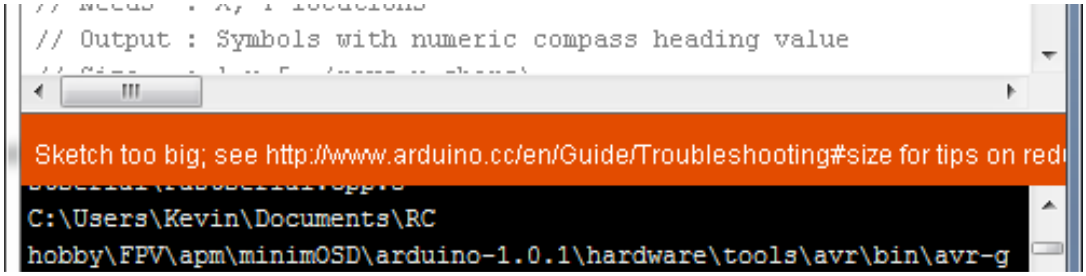
Notice how the addresses matches up with the “CR” and “m/s” images in the Max tool. Say we don’t want to have the “CR”. Instead of editing out the character in the charset file, we can use the blank space character “0x00” like this:

```
    osd.printf("%c%3.0f%c",0x00, (osd_climb), 0x88);
```

Now say we also want climb rate in mph instead of m/s. We can add math inside the print function, and use the mph character like this:

```
    osd.printf("%c%3.0f%c",0x00, (osd_climb)*2.24, 0xFB);
```

We can get obscure and use mm/hr with the right conversion factor, and editing the “0x88” character with the Max tool. Now that we’re done with our mod, we click the compile button. Eventually with we do too much and compiling fails with “Sketch too big”



Fear not, there are more than likely some panels you would never end up using. You can remove them to make room to do more with your hex file. A good for saving room is the heading rose, since there is also a heading angle panel that can serve the purpose. So we’ll remove the rose for our example of saving bits of space. First comment out all the active lines in the panel with “//” as shown below:

```
/* ************************************************************ */
// Panel  : panRose
// Needs  : X, Y locations
// Output : a dynamic compass rose that changes along the heading information
// Size   : 2 x 13  (rows x chars)
// Staus  : done
//void panRose(int first_col, int first_line){
```

```
//osd.setPanel(first_col, first_line);
//osd.openPanel();
   //osd_heading  = osd_yaw;
   //if(osd_yaw < 0) osd_heading = 360 + osd_yaw;
// osd.printf("%s|%c%s%c", "\x20\xc0\xc0\xc0\xc0\xc0\xc7\xc0\xc0\xc0\xc0\xc0\x20", 0xd0, buf_show, 0xd1);
//osd.closePanel();
//}
```

Notice the name of the official panel name highlighted on top, “panRose”. Now scroll up near the top of OSD_Panels where several “if” statements are located. Find the line with “panRose” in it, and comment it out:

```
//  if(ISb(panel,Rose_BIT)) panRose(panRose_XY[0][panel], panRose_XY[1][panel]);       //13x3
```

Now chances are that saved enough room for some more modifications. You can do this to remove any panel you don’t need. Some panels are larger than others… have fun experimenting with flight!