

TheOpenCockpit Airduino 4.3 Version 1.0 Beta

The Airduino project is an open source Arduino Based experimental aircraft PFD. It is intended for experimental and educational use. It is hoped that with farther development, it can be useful as low cost backup instrumentation for experimental aircraft. This manual describes the construction and software for version 1.0 beta.



The photograph above shows the second prototype Airduino with the Yost Labs AHRS and a 9 volt battery.

Release Status: This project is in Beta Release. You are welcome to play with it, but it is not yet considered reliable enough for the general public.

Known Issues: Please see the dynamic issue list on Github at the following link: <https://github.com/the-open-cockpit/Airduino/issues>. If you discover new issues, they should be posted here.

Warnings and Disclaimers: Any time an aircraft flies, there are certain risks involved. These risks are multiplied when a person decides to fly an experimental aircraft. These risks are multiplied again when a person decides to fly an experimental aircraft that has experimental avionics.

The experimental PFD described in this booklet is highly experimental. It should not be used in flyable aircraft except for the purpose of further development or as a backup in case normal systems fail. In either case, full redundant systems of a non-experimental nature should be provided in any aircraft an Arduino is installed in. These systems should be regarded as more reliable than the Arduino. Should they communicate information that does not agree with the Arduino, the other systems should be considered accurate, and the Arduino considered inaccurate.

The Arduino is experimental, and may not be installed in certificated aircraft.

Any and all liability of any operation of any Arduino instrument shall be fully upon the operator of the instrument. In no case shall the creators of the Arduino accept any liability.

License: This software is licensed under the BSD 3 Clause license as follows:

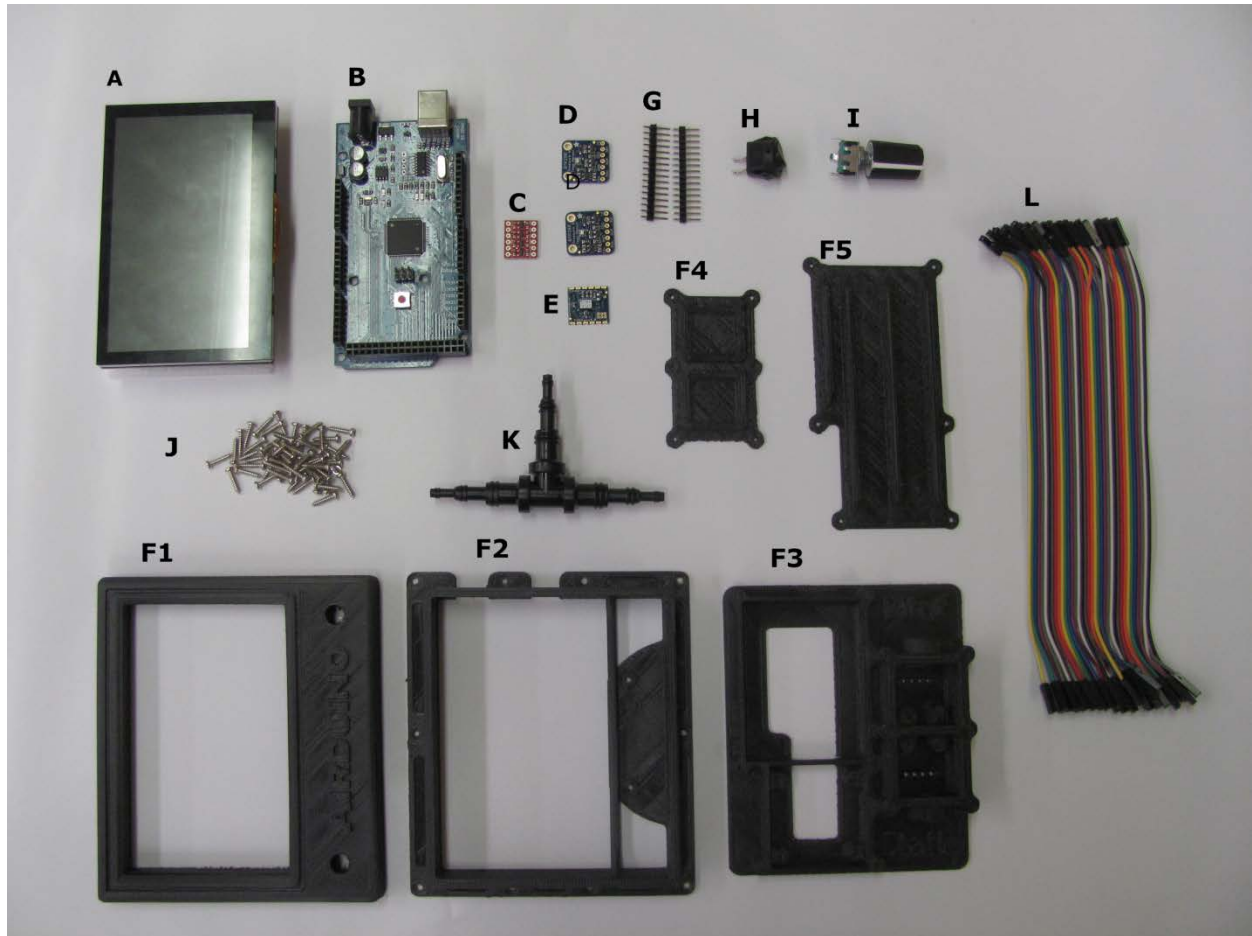
Copyright 2019, Don Morris and Chongwen Chen

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Hardware Materials List:



- A. Newhaven 4.3" Display Sunlight Readable Screen Shield :
<https://www.mouser.com/ProductDetail/Newhaven-Display/NHD-43RTP-SHIELD-N?qs=sGAEpiMZZMtr3BLCQWqPFfcy2oxL2wNkBuCdyCj4tQESnikW1qfhbg%3d%3d>
- B. Arduino Mega 2560 Rev 3: (Note – cheaper clones may or may not work)
<https://www.mouser.com/ProductDetail/Arduino/A000067?qs=sGAEpiMZZMt0re6d%252b2Rx9v%252bc%252bQElaOW9>
- C. Boards Logic Level Converter: (should be I2C compliant)
<https://www.mouser.com/ProductDetail/SparkFun/BOB-12009?qs=%2fha2pyFadugTiatgUFVhGL8Ys07VBb%2fA28as344qKEE9feJyPPQRLA%3d%3d>
- D. BMP280 Pressure Sensor: (need 2 sensors, one for static and one for pitot)
<https://www.mouser.com/ProductDetail/Adafruit/2651?qs=sGAEpiMZZMvxSQPygxWTpQkikPvcfyGhD5IMKcncvnE%3d>
- E. 3-Space Nano Embedded Sensor:
<https://yostlabs.com/product/3-space-nano-embedded/>

F1-F5. 3D Print Case: The instruction for printing out the case is published on the thingiverse.com website. Please visit the link below and follow the detailed instruction.

<https://www.thingiverse.com/thing:3246908> . The case should be printed in ABS plastic.

G. Header Pin (optional).

H. Power Button: SPST on - off switch, use any switch you think is appropriate.

I. Encoder (selector knob):

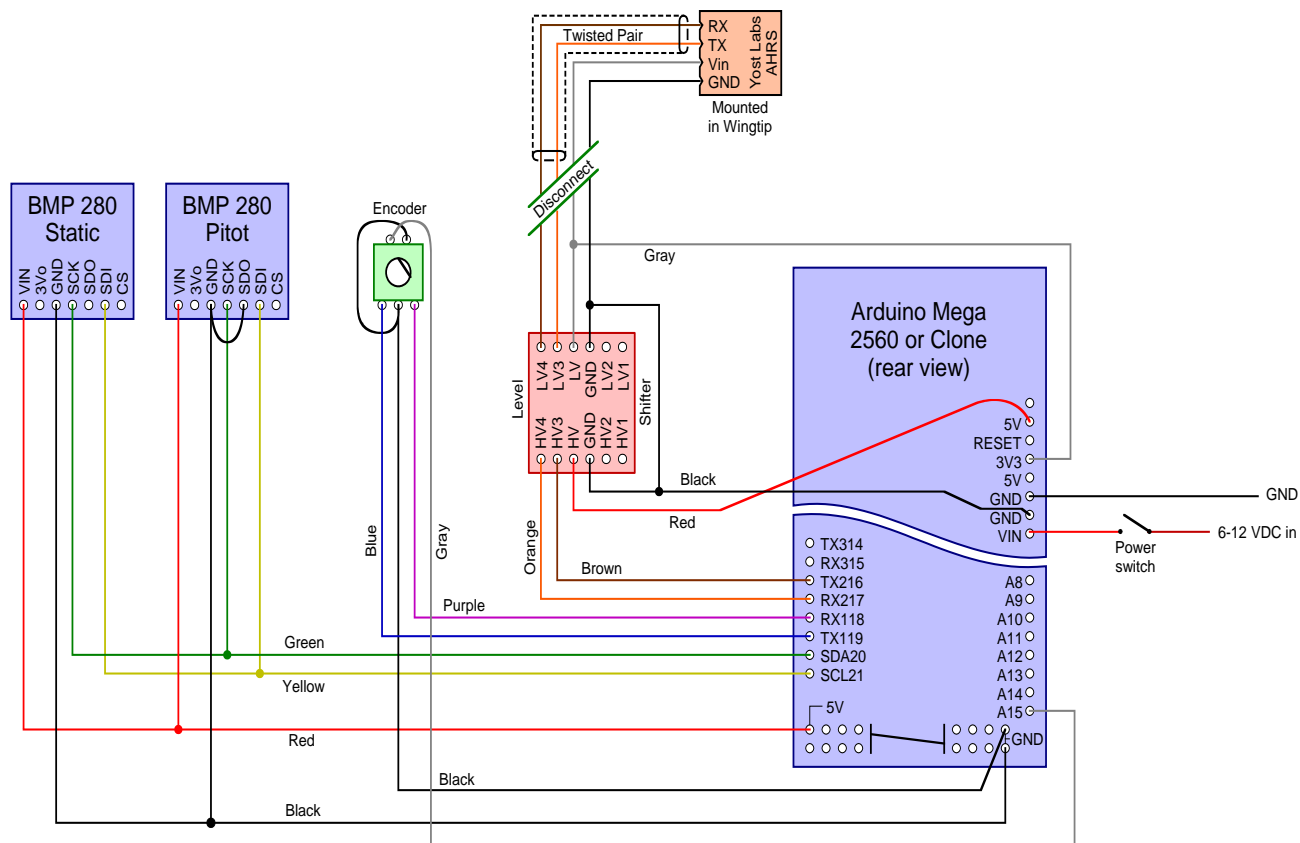
<https://www.mouser.com/ProductDetail/BI-Technologies-TT-Electronics/EN11-VNB1BQ15?qs=%2fha2pyFadug3gL3ObFWUhXGm41d47hiWle0IFPxnvmkhknsfGXR%2f9A%3d%3d>

J. #2 Screws.

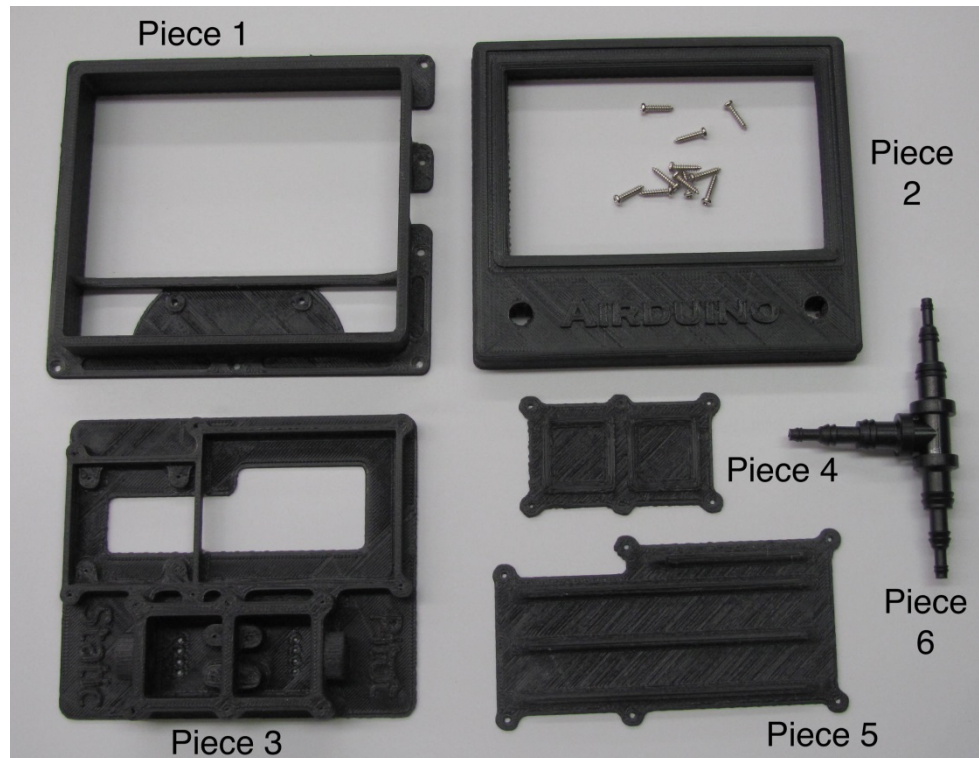
L. Header wires

Hardware Assembly:

Electrical Schematic



Case Parts:

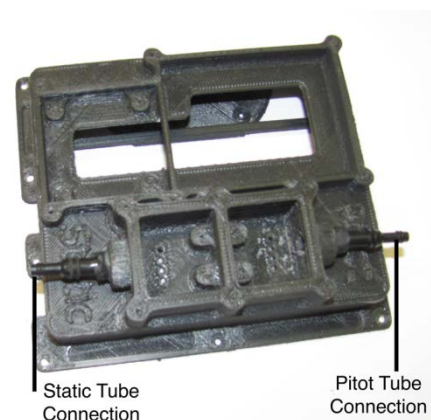


Step 1: Glue Case Parts 1, 3, and 6 Together

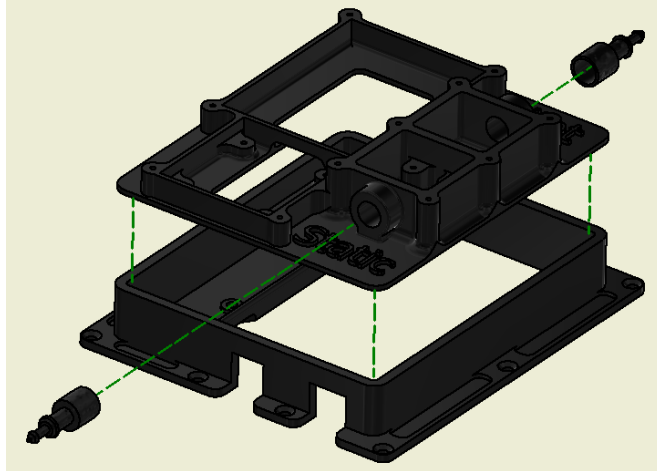
The case was printed in several parts to minimize the need for supports and the corresponding rough finish. It should have been printed in ABS plastic. You will glue it together using ABS glue. We recommend you use an all-purpose plumbing cement that includes ABS. The Oatey cement in the next photo works very well.



Parts before gluing (note cut Tee fitting)



Parts after gluing



You will use the glue to attach the parts of the case together as above. Begin by removing the support material from the Pitot and Static port holes as shown below. We find that a step drill works well for smoothing the plastic prior to gluing. Next, cut two of the ends off of the Tee – piece 6. These ends must be sanded carefully so that they can be inserted into the pitot and static port holes. Once they can be pushed into the holes, they are ready to be glued in place. Do not make them too loose, or they will not seal properly. Too strong of a press fit will make the case layers delaminate.



Preparing Case for Hose Fittings



Sanding Hose Fittings to go in Case

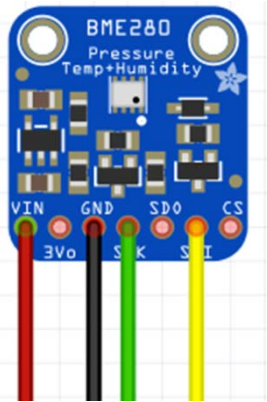
Continue by gluing piece 3 to piece 1 as shown in the photographs and diagrams. The plumbing glue sets relatively quickly.

Step 2: Attach wires to the BMP-280 sensors.

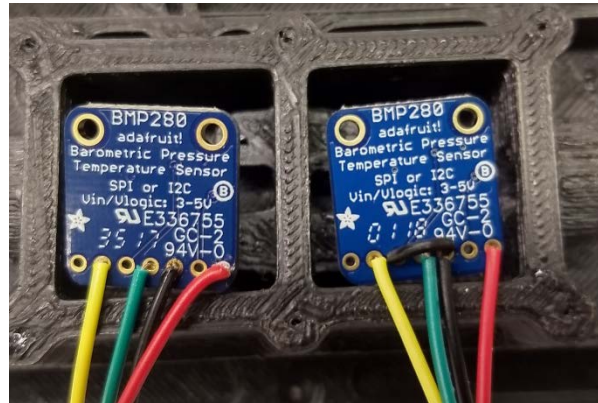
The Adafruit BMP280 sensors have built in logic level shifting. The same sensor is used for the Pitot and the Static. The only difference between the two is the additional jumper wire between

GND and SDO on the Static Sensor. This jumper changes the I2C address on the chip, so that they can each be queried separately by the Arduino.

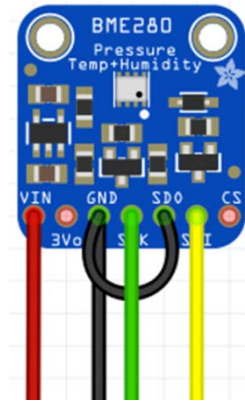
If you are using jumper wires as per the material list, separate the ends from each of the wires you will need. Solder the Red, Green, Yellow, and Black wires to each of the sensors as shown.



Front of Static Chip



View of Actual Chips and Wires



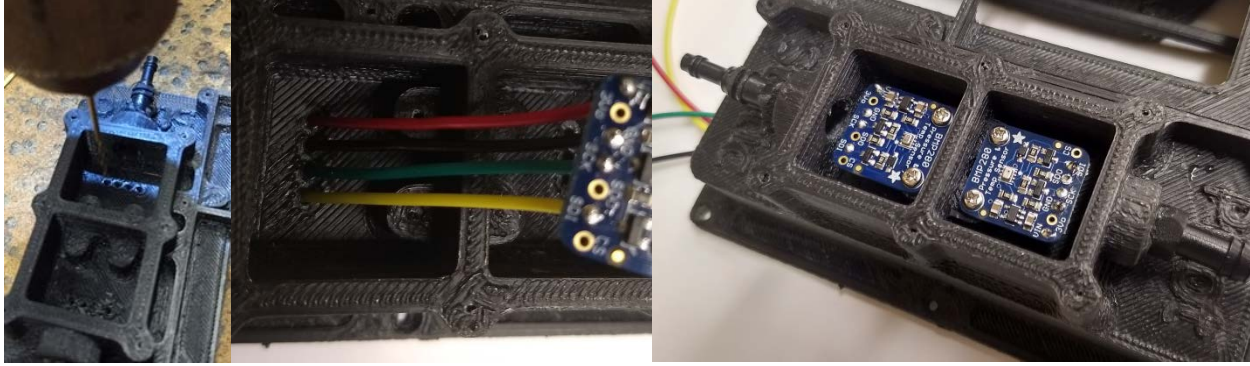
Front of Pitot Chip

Note that the wires are soldered to the pads leaving to the rear of the chips. This allows the chips to be mounted in the pressure compartments in the case.

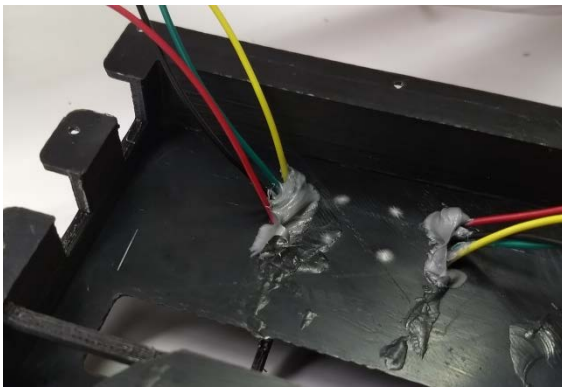
Step 3: Install the Pitot/Static Boards in their Compartments

Both the Pitot and the Static BMP-280's must be in sealed compartments in order to sense pitot and static pressure on the airplane. These are used to calculate airspeed, vertical speed, and altitude. These will also be an essential part of other calculations as the instrument becomes more capable.

Begin by drilling out the 8 wire holes (4 per chamber) using a small drill bit that is just larger than your wires. These holes were built into the printed case, but likely are not the correct size.



Pull each of the four wires from the BMP-280 boards through the holes in order as shown. Remember that the Pitot board has the extra jumper. Make sure you get the correct chip in the correct chamber. Gently screw the breakout boards into place on the provided pads, also as shown.



Silicone sealant on reverse side of wires



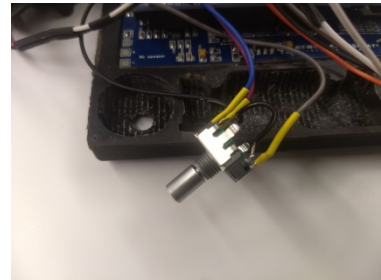
Silicone sealant and chamber lid

After the breakout boards are screwed in position, use silicone sealant to seal the backs of the wires on the reverse side of the case. We pulled the wires to one side and sealed them, then pulled the wires the opposite direction and reapplied the sealant. While the pressure differentials across the case are quite small, it is important that there be no leakage.

After sealing the wires, you may use more of the same silicone sealant and screws to seal the pressure chamber lid (Piece 4 in original diagram) to the top. Apply more sealant than you think will be necessary. However, do not apply so much sealant that it will squeeze out and block the pressure sensor on the breakout boards below. Any leaks here will be very difficult to diagnose, but will prevent reliable pitot-static operation.

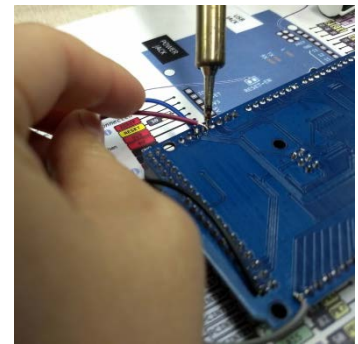
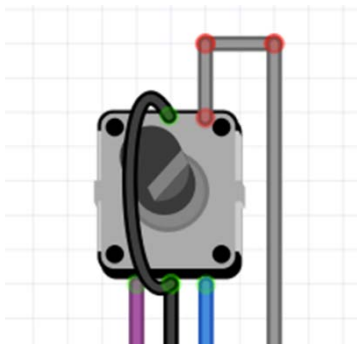
Step 4: Wire up the Power Switch and Multi-Select Knob

Begin by fitting your multi-select knob and power switch to the case front (Piece 2). You will likely need to change the size of the holes in the printed case in order for these to fit the case. We found that a step drill worked well for this.



We chose to solder wires to the power switch while it was mounted in the case, but to solder wires to the multi-selector while it was outside of the case. The power switch really needs no explanation, as a red wire between it and the back pad of the V-in pin of the Arduino are all that is necessary. The other side will simply hang out the back of the case and be connected to a 6-12 volt input voltage.

The multi-selector is a bit more complicated. It features a rotary encoder with a momentary NO pushbutton switch. It a ground line attaches to the common pin, and three data lines go back to the Arduino.



We used black for the ground line, which is used to pull the data pins on the Arduino low. The black wire is attached to the center pin of the encoder side (3 pins), and then jumpered to either pin of the pushbutton side (2 pins). We did not both with heat shrink on the ground wires. The other wires were attached as shown, using heat shrink tubing to ensure that no signals could short to ground. All four wires run to the solder pads on the rear of the Arduino for the proper pins.

Step 5 – Wire in the Logic Level Shifter and the ARHS Board

The Yost Labs ARHS board operates at 3.3 volts, while the Arduion MEGA operates at 5 volts. This means that a logic level shifter is essential in order to avoid board damage. It is our

intention that the AHRS board be remotely mounted in a magnetically neutral location, such as a wingtip. High speed serial data between the Arduino and the AHRS should travel on appropriate twisted pair shielded wire. In addition, an appropriate disconnect plug is highly recommended. We have not called out any particular plug, but would recommend a high quality aviation or automotive style connector.



Note: The RX on the AHRS needs to connect to the TX on the Arduino and the TX on the AHRS needs to connect to the RX on the Arduino. In the overall wiring diagram, we switched the color of the wires at the logic level converter. In this diagram, we did not, choosing to connect the LV Orange wire to the AHRS TX.

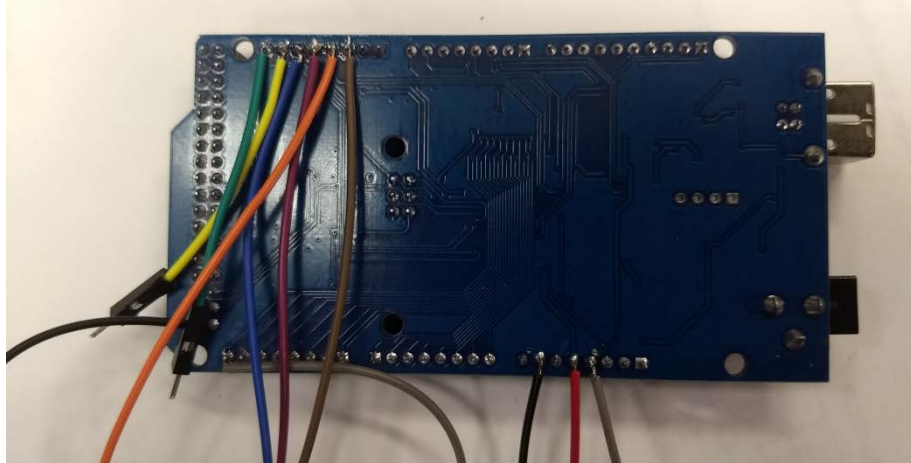
The logic level converter has two channels. We will only be utilizing one. We have specified orange and brown wires for UART serial communication. These data wires are soldered between the High Voltage side of the breakout board and the Arduino. An additional black ground wire is also used between the board and the Arduino. The ground wire is farther jumpered across to the ground on the LV side of the breakout board (not shown, behind board), where it continues on to the disconnect plug (also not shown). All wires attached to the Arduino are attached to the rear solder pads.

The Arduino board has a built-in 3.3 Volt power supply, and it will be used to power the AHRS. We used a grey wire to go between the LV side of the board and the 3V3 pin on the back of the Arduino. It then continues on to the disconnect plug (not shown).

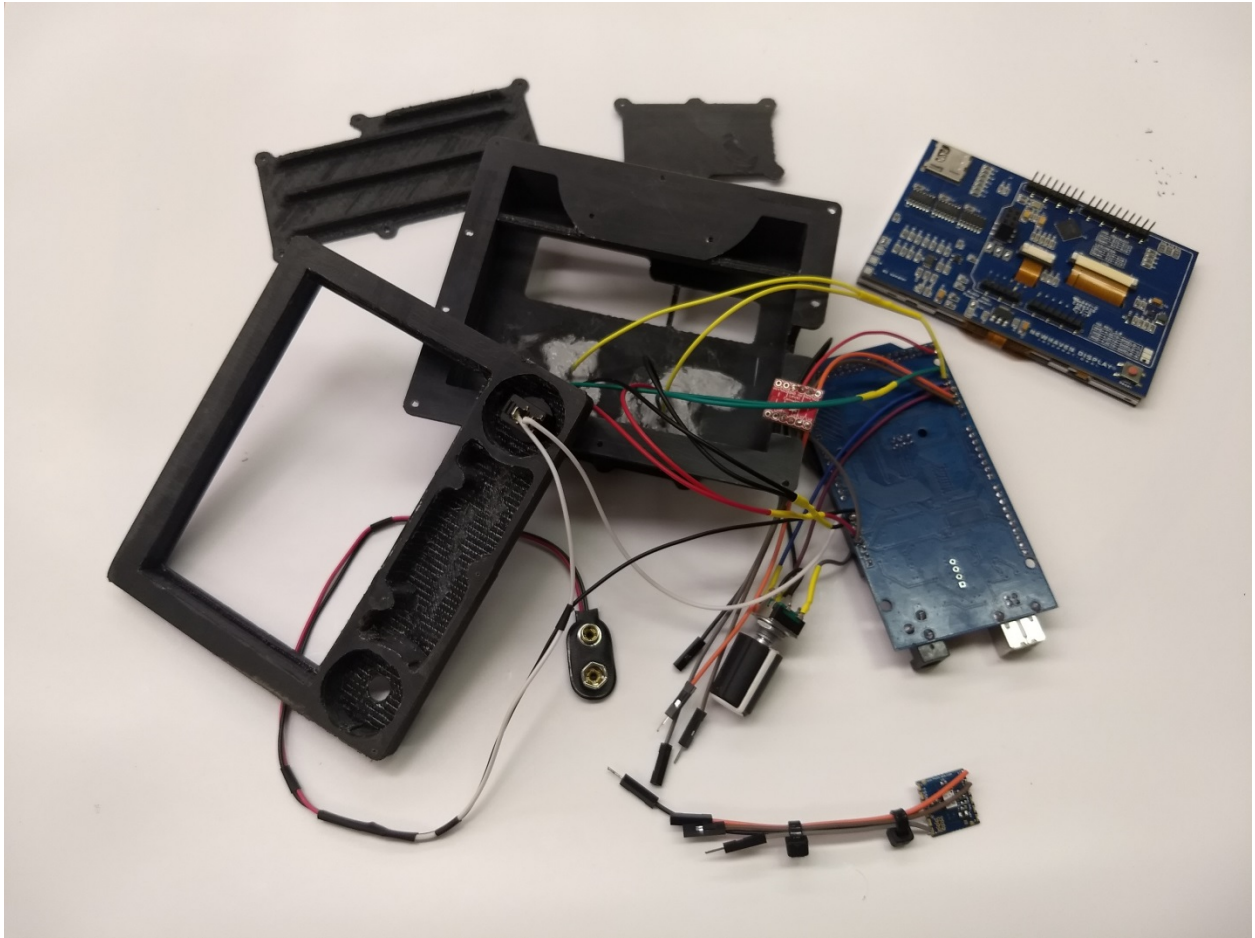


Since our demo board was not going to be installed in a real aircraft, we did not use a twisted pair of wires or a high quality disconnect. We just used the male/female headed wire ends for our disconnect. This proved fine for our experimentation, but is highly inadequate if mounting in any type of test bed.

It would probably be wise to provide some type of housing for the AHRS board, and perhaps another disconnect plug at this end as well.

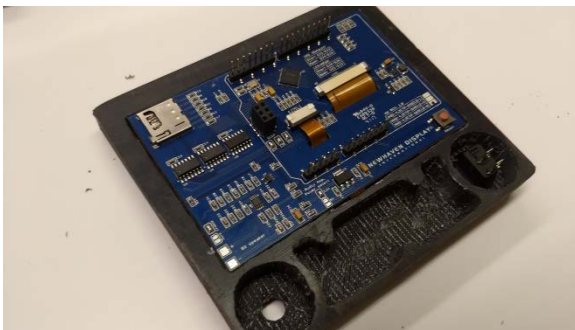


Wires by color on the back of the Arduino MEGA.

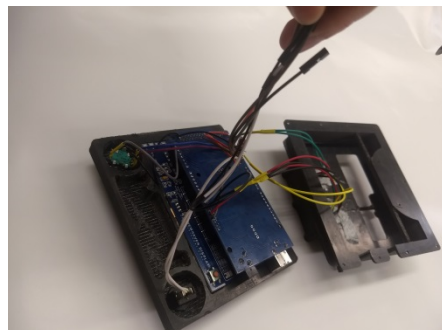


Step 6: Putting it all together.

At this point, it is time to put everything together. The above picture shows everything to date, with the power wire hooked to a 9 volt battery connector. Yours may look different from this, depending on your chose voltage source.



Fit the screen into the back of the bezel (Piece 2).



Fit the Arduino MEGA onto the back of the screen.

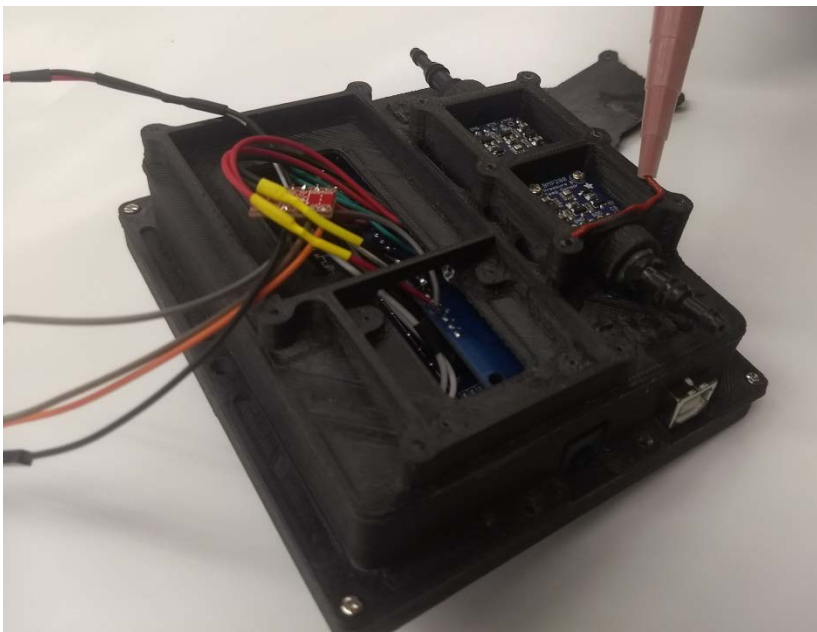
Splice the two yellow wires to the Pitot and Static breakouts to a single yellow wire that attaches to the shown pin on the Arduino. Do the same with the two greens, two red, and two blacks. See below. Note the heat shrink over the solder splices.



Place a large piece of heat shrink over the logic level shifter (not shown). Alternately, wrap the logic level shifter in electrical tape.

Carefully pull the power wires, the lines to the logic level shifter, and the lines to the AHRS disconnect through the openings in the rear of the case.

Fit the entire package together, and screw everything in place.



Note the slightly different sequence in photos, where we still had to silicone down the chamber top. We used four screws in the corners, which allowed us to use the remaining screws to mount the Arduino in a panel.

As previously noted, putting heat shrink over the logic level shifter would be a good idea. Wrapping in electrical tape would also help, but not as much.

A better mounting system would be nice.

Software and Installation:

Required Software:

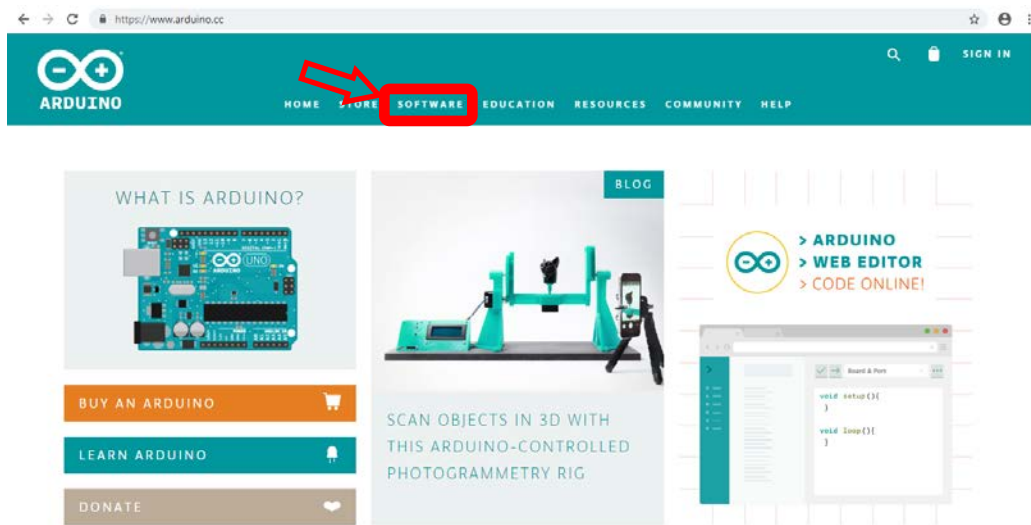
These programs may be downloaded for free. Follow the links from www.theopencockpit.org.

1. Arduino IDE. Recommend latest version. Download and install on your computer as per instructions on the Arduino web site. Provided by the Arduino. Web site www.arduino.cc.
2. Airduino Source Code. Program file developed in Arduino IDE. This is the source code for the project. Web site www.theopencockpit.org. Will direct you to www.github.com for download.
3. Required Libraries. These are provided by the board manufacturers to allow the Arduino to talk with the chips. These are in individual packages that must be installed in Arduino before. If you do not install these, the software will not run.
 - a. Adafruit Sensor Library
 - b. Adafruit BMP-280 Library
 - c. Display Shield Library
 - d. Optical Encoder Library

Download and Installation:

Many of the steps in this section depend on your computer system. This section will work you through the steps on a Windows based computer system, but this can also be accomplished on a Linux or a Mac computer following instructions that are available on Arduino forums. This process works as of April, 2019. If the web site or software has changed, please follow the directions located on the web site.

1. Download the Arduino IDE and install it on your computer.



- a. Go to Arduino.cc, click on the software section, and select “Downloads.”

Download the Arduino IDE



ARDUINO 1.8.9

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. Refer to the [Getting Started](#) page for installation instructions.

Windows Installer, for Windows XP and up
Windows ZIP file for non admin install

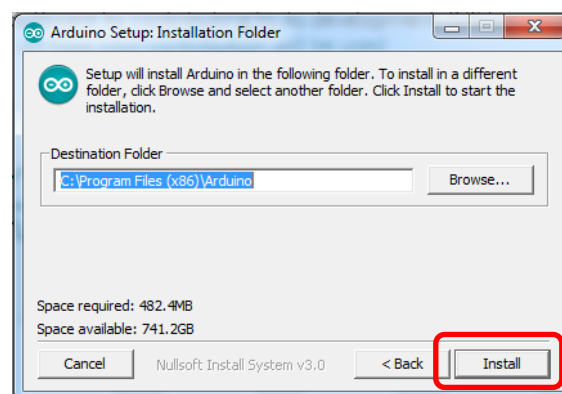
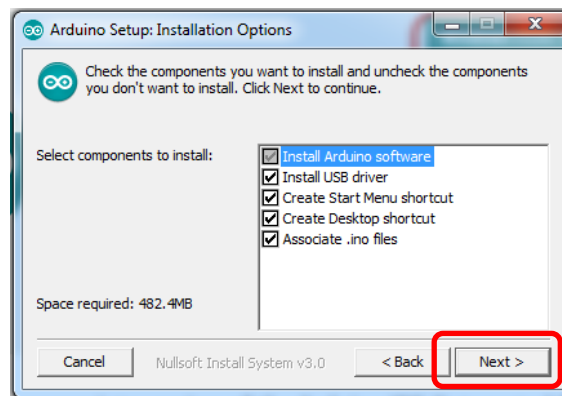
Windows app Requires Win 8.1 or 10
[Get](#)

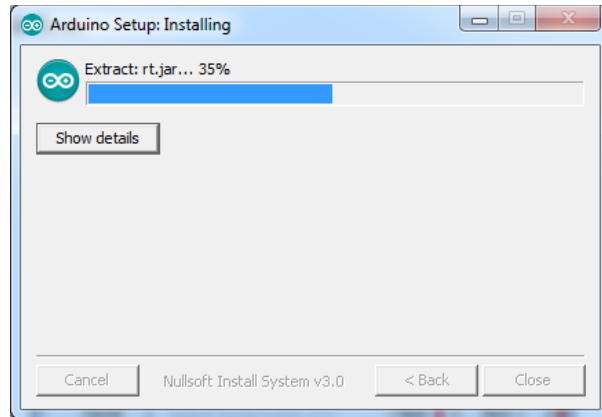
Mac OS X 10.8 Mountain Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

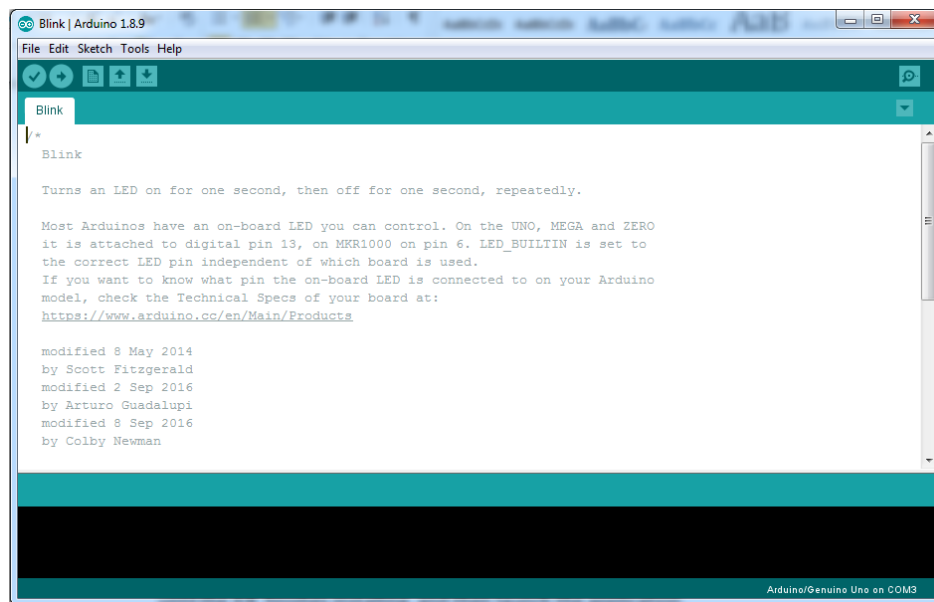
[Release Notes](#)
[Source Code](#)
[Checksums \(sha512\)](#)

- Select the appropriate version of the Arduino IDE (Integrated Development Environment). Click on it, and it should download.
- Open the file, and give it permission to install on your computer. The default options should work fine.

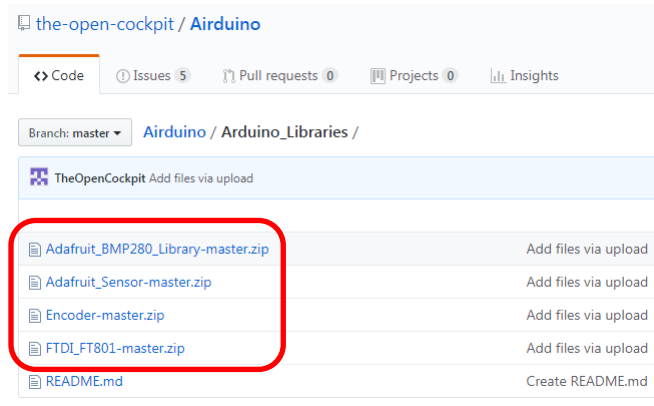




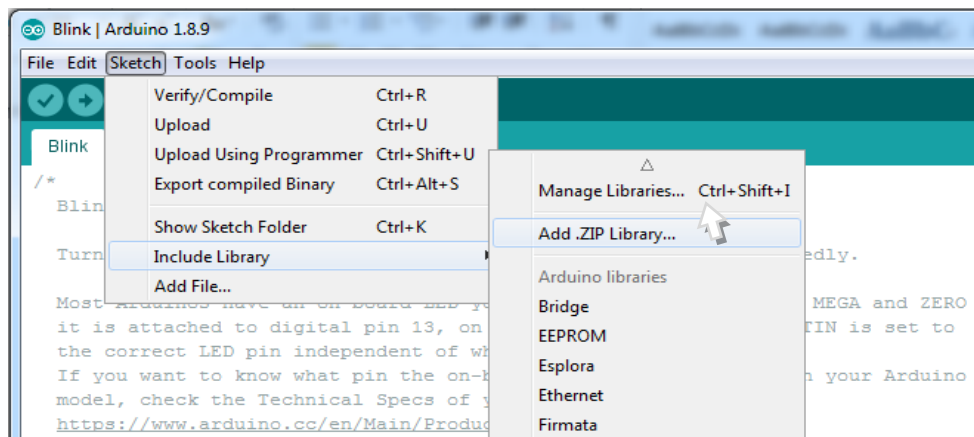
- d. You may have to give the program permission to install several files and drivers. Wait until the IDE finishes installing.
-
2. Install the four required libraries in Arduino.
 - a. Launch the Arduino IDE.



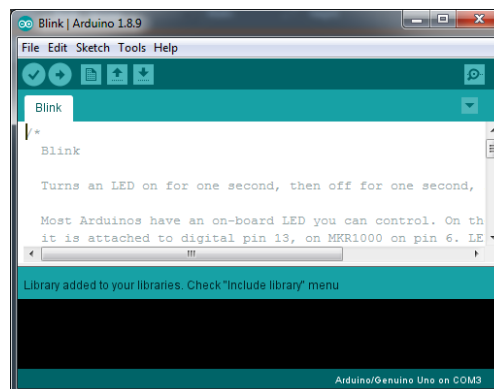
- b. Download the four zip files from the Arduino_Libraries directory of Github. Make sure you know where you save them on your computer. Do not unzip them.



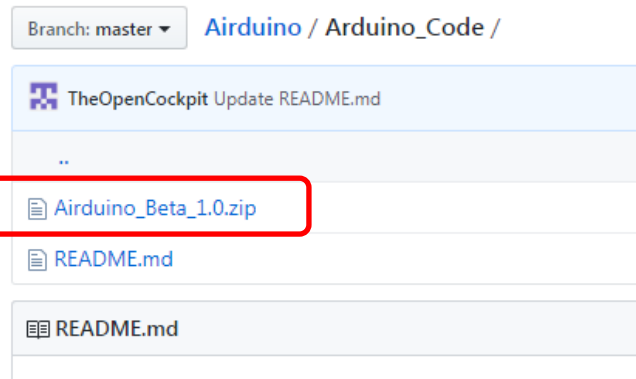
- c. Back in the Arduino IDE, select “Sketch” and then “Include Library” and finally “Add .ZIP Library.”



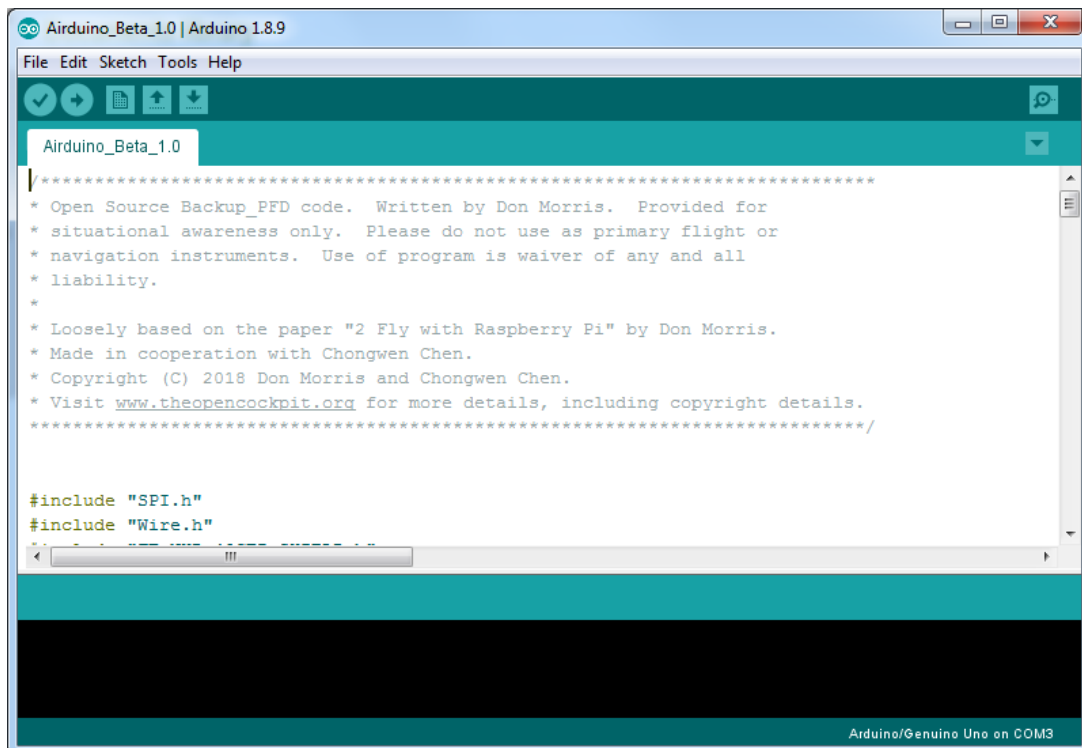
- d. This will bring up a window, and you can navigate to wherever you saved the four libraries. Select one of the libraries, and the program should install it for you. The screen below shows how my computer looks when it is successful.



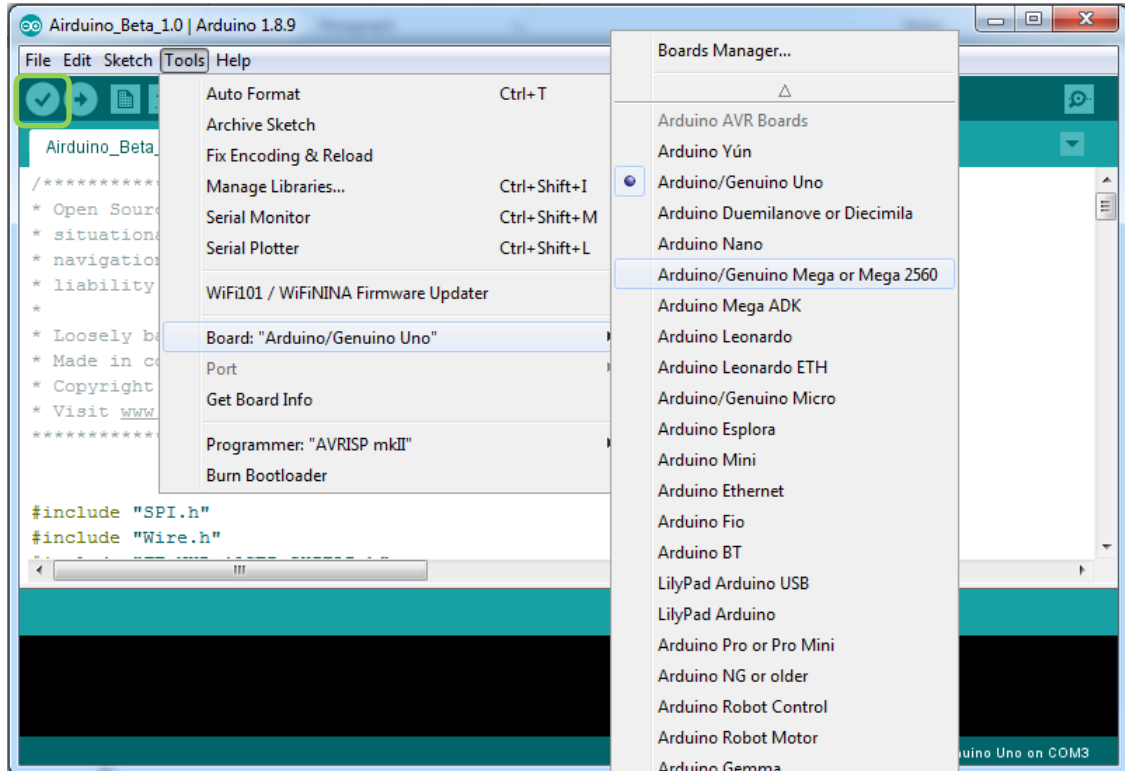
- e. Repeat steps “F” and “G” until all four libraries are installed.
3. Open the Airduino program in your Arduino IDE.
 - a. Download the Arduino code from Github. Note where you save it. This time, you will want to unzip it.



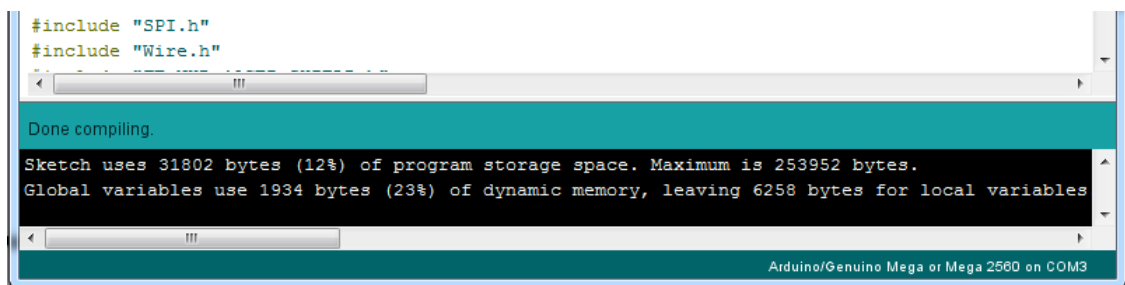
- b. In the Arduino IDE, click “File” and then “Open.” Navigate to where you unzipped the folder in the last step, and open the file. If you are successful, you should see the following.



- c. Switch the type of board to “MEGA 2560” – the kind of board we are using in the Arduino. Go to “Tools” and then “Board” and finally select “Arduino/Genuino Mega or Mega 2560.” (Note that if you have used an off-brand MEGA 2560 board, you may have to use a different board setting here.)

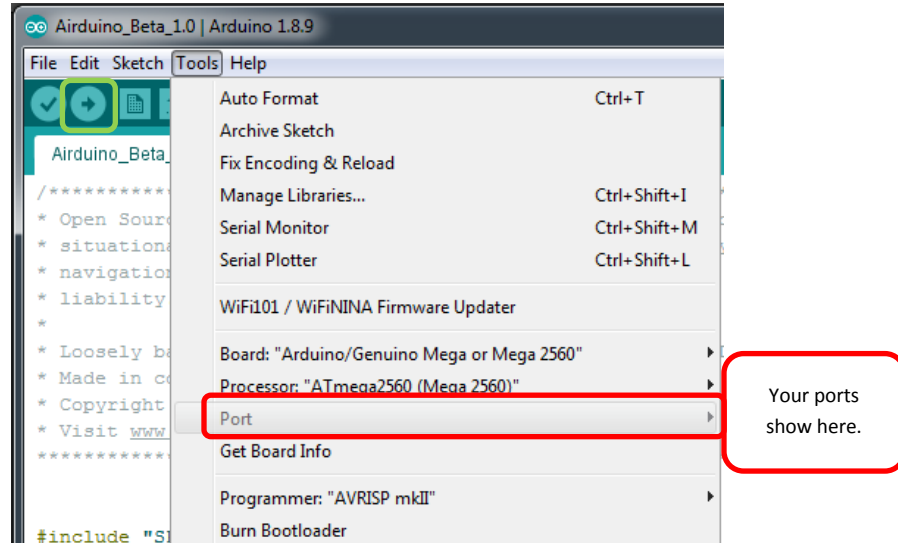


- d. Finally, test your installation by clicking on the Check Mark on the upper left side of the program (shown in the above picture in the green box). If the installation is correct, you will see text in the black box at the bottom. It will take a few moments, but you should get the following:

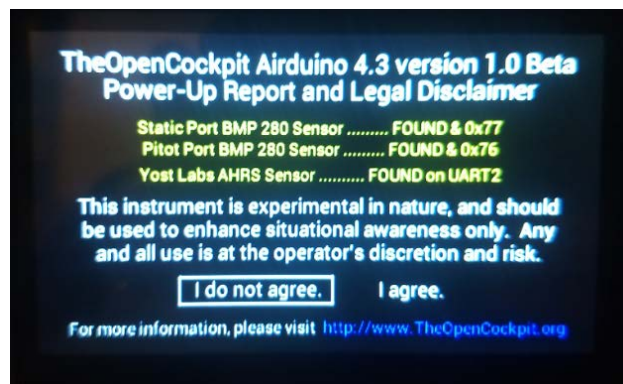
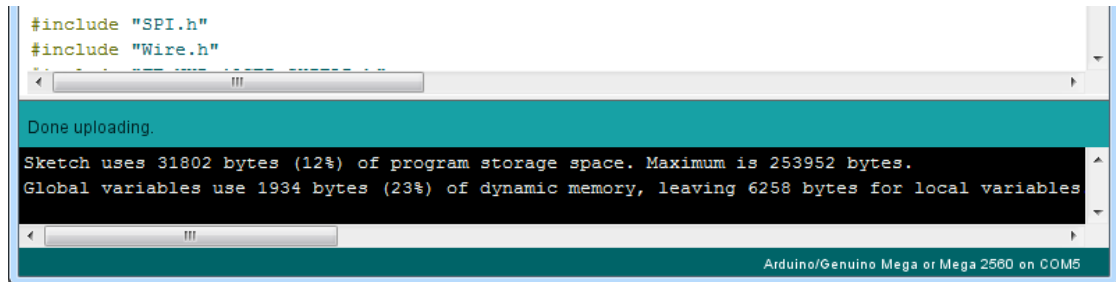


4. Upload the program to the Arduino unit you built.
- a. Plug the unit into a USB port on your computer using a standard USB cable. Allow the computer to install the proper serial port.

- b. Select the appropriate port on the “Tools” “Port” menu shown in red below.
- c. Click on the “Upload” button that is shown in green below.



- d. It will take a little time to compile the program like it did before, and then it will upload the program to the board. If it is successful, you will see the something like what is below. You Arduino will turn off, and then turn on again, showing the lower screen.

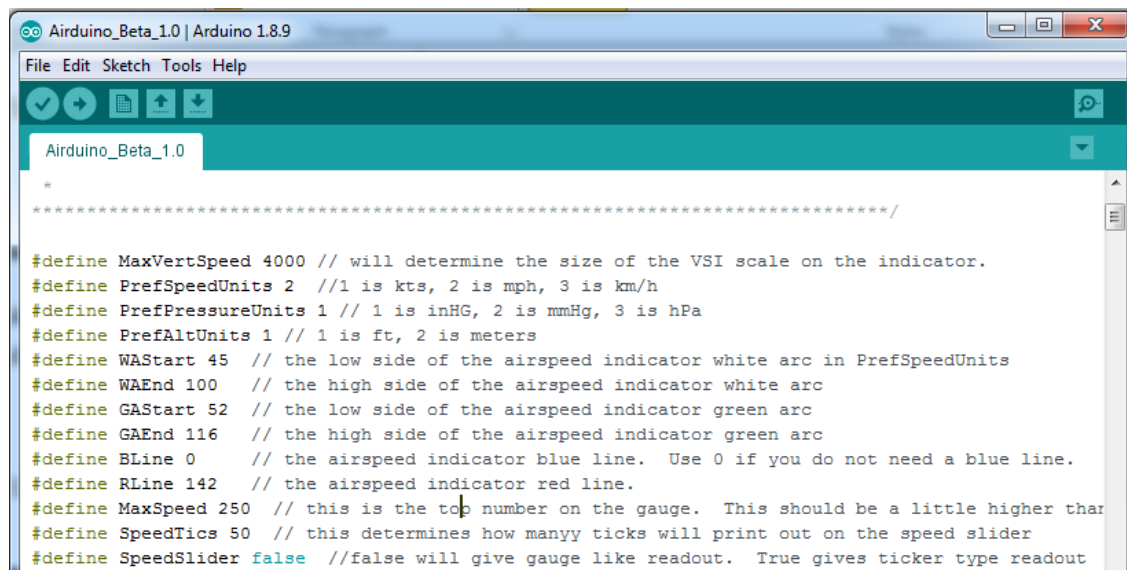


Note that you must agree to the disclaimer to continue. Turn the multiselect knob, and press “I agree.” If you have trouble selecting the “I agree” position, you may need to change the “`#define ECPD 2` // number of encoder clicks per detent. Probably 1, 2, or 4.” as detailed in the next section. Also note that either of the static port BMP 280 sensors or the Yost Labs AHRS sensor cannot be found, these will show up in red. This probably means you have a wiring error.

5. Configure the software using the define statements as detailed in the next section. After you have configured the software in the Arduino IDE, you will need to re-upload the software as in section 4.

Software Configuration:

As is typical of Arduino applications, define statements at the beginning of the code are used to define the options you wish to use. Change the numbers by each option only. In the case of a true/false option, type true or false in all small letters. If the word is recognized, the color will switch as shown below next to “SpeedSlider.”



```

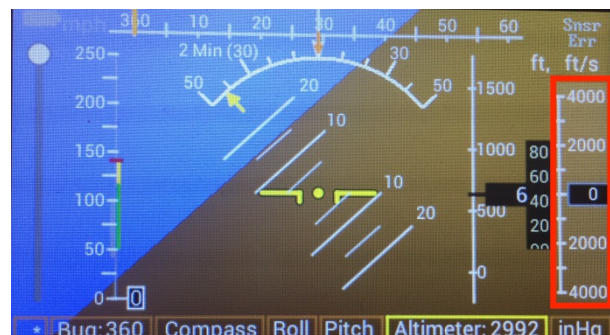
Airduino_Beta_1.0 | Arduino 1.8.9
File Edit Sketch Tools Help

Airduino_Beta_1.0

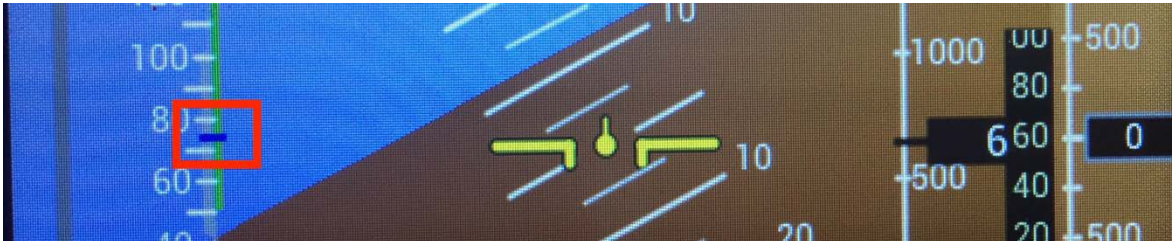
*
***** /

#define MaxVertSpeed 4000 // will determine the size of the VSI scale on the indicator.
#define PrefSpeedUnits 2 //1 is kts, 2 is mph, 3 is km/h
#define PrefPressureUnits 1 // 1 is inHG, 2 is mmHg, 3 is hPa
#define PrefAltUnits 1 // 1 is ft, 2 is meters
#define WStart 45 // the low side of the airspeed indicator white arc in PrefSpeedUnits
#define WEnd 100 // the high side of the airspeed indicator white arc
#define GStart 52 // the low side of the airspeed indicator green arc
#define GEnd 116 // the high side of the airspeed indicator green arc
#define BLine 0 // the airspeed indicator blue line. Use 0 if you do not need a blue line.
#define RLine 142 // the airspeed indicator red line.
#define MaxSpeed 250 // this is the top number on the gauge. This should be a little higher than
#define SpeedTicks 50 // this determines how many ticks will print out on the speed slider
#define SpeedSlider false //false will give gauge like readout. True gives ticker type readout
  
```

1. `#define MaxVertSpeed`. Number determines top and bottom of VSI scale displayed. You may utilize any number you like. 1000 below on left. 4000 shown above and below on the right.



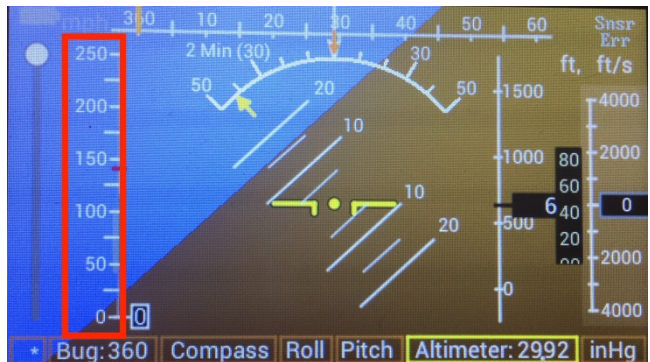
2. `#define BLine 74` // the airspeed indicator blue line. Use 0 if you do not need a blue line.



3. `#define MaxSpeed`. Number determines the top number on the gauge. This should be a little higher than your red line. 160 and 250 shown below.



4. `#define SpeedTics`. Number determines how far apart the numbers are on the speed slider. Can be any number, but be reasonable. Default 20. 50 also recommended.



5. `#define BottomWhiteArc`. Number chooses the bottom of the white arc.
6. `#define TopWhiteArc`. Number chooses the top of the white arc.
7. `#define BottomGreenArc`. Number chooses the bottom of the green arc.
8. `#define TopGreenArc`. Number chooses the top of the green arc and bottom of the yellow arc.
9. `#define RedLine`. Number chooses the top of the yellow arc and the red line.

10. #define HeadingNum true (or false) // Display the heading in a black box in the center of the DG strip.

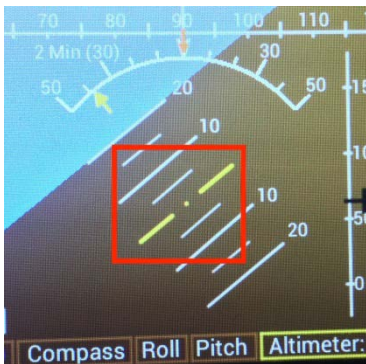


False (no black box)



True (Black box with text heading)

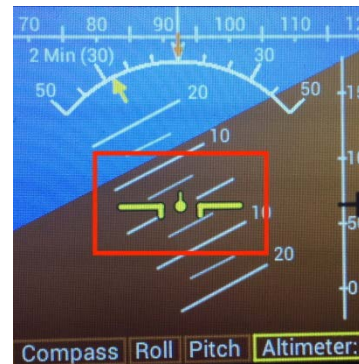
11. #define CenterDisplayType 3 // 1 is yellow lines with the horizon. 2 Yellow lines stay level. 3 gives both.



Type 1



Type 2



Type 3

12. #define dynamicsky true // false does not change the display color based on pitch. true does.



True: as pitch goes up, the background color (blue color) gets darker and darker. Compared to the blue in the picture on the right.



False: Sky remains color shown in this photo regardless of pitch.

13. #define dynamicground true // false does not change the display color based on pitch. true does.

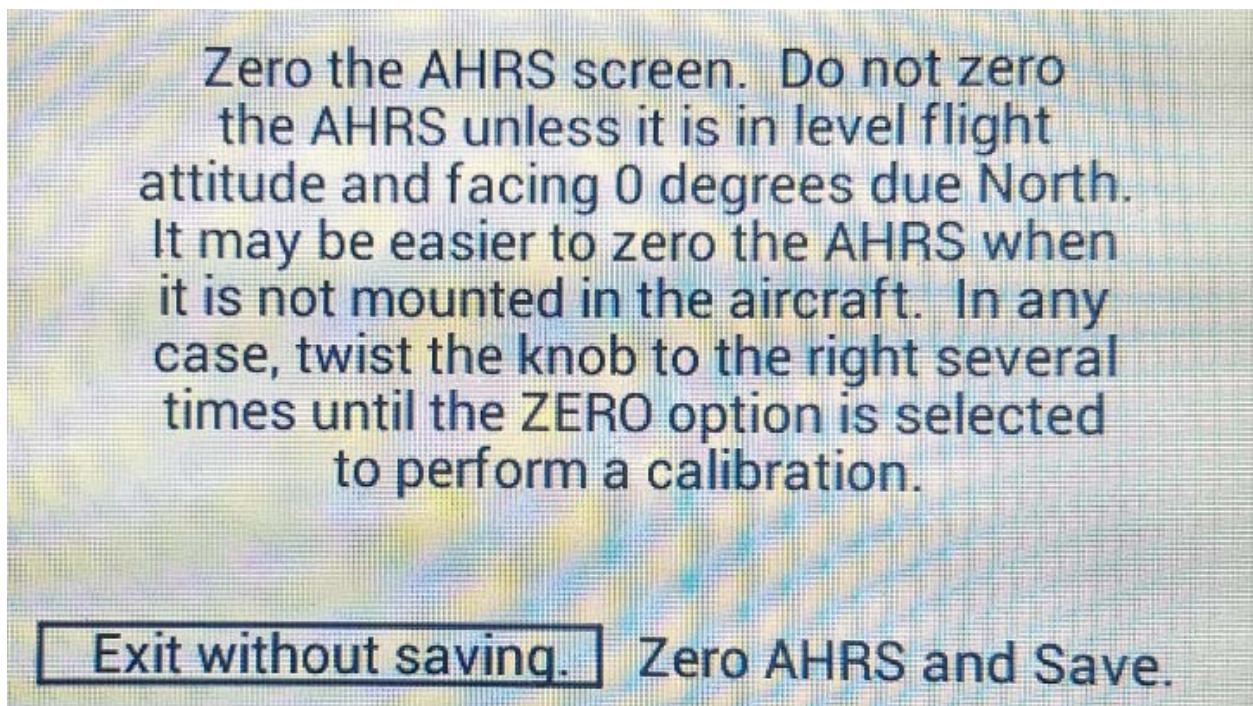


True: pitch forward, the ground color (brown color shown on right) becomes increasingly red.



False: as pitch goes down, the ground color (brown color shown here) does not change

Sensor Calibration:



This screen will zero the ARHS chip in whatever location it is. It does not (yet) save the new zero position after the power is turned off, but this will be added soon. To reach this screen, turn off the unit. Hold the selector knob down while turning on the power. You should let up on the knob as soon as this screen appears. In order to move the cursor over to "Zero AHRS and Save," twist the knob in that direction for several turns. Then press the select knob again to perform the zero.

About the Authors:

Don Morris is a professor of Aviation Technologies at Southern Illinois University, Carbondale. He is an aviation buff, with a love for experimental aviation. He is also heavily involved in the EAA chapter 277.

Chongwen (Grace) Chen is completing her BS in Aviation Technologies at SIUC in the Spring of 2019. Her involvement in this project was for an honors thesis. She is also wrapping up a commercial helicopter certificate.

Joining in the Development:

If you are interested in contributing to this project, please contact don@theopencockpit.com. There are many areas where contributions would be helpful. These include additional software development and documentation.