

CSCE 4925: Project Aero

Arduino Guide

By: Alyssa Thurston

Project Manager: Alyssa Thurston

Instructor: Professor Keathly

Client: Dan Minshew and Kyle Tayle, Denton Techmill

Date: 5 May 2018

Status: Final Draft

Table of Contents

| | |
|--|-----------|
| Table of Contents | 2 |
| Revisions | 3 |
| 1 Materials and Software Needs | 4 |
| 1.1 Materials | 4 |
| 1.2 Software Needs | 4 |
| 2 Building Your Node | 6 |
| 3 Troubleshooting | 17 |
| 3.1.0 Code Won't Upload | 17 |
| 3.1.1 Nothing Shows Up on the Serial Monitor | 17 |

Revisions

Below is a list of any revisions made to this document.

| Date | Description of Change Made | Person Making Change |
|----------|----------------------------|----------------------|
| 5/5/2018 | Draft Created | Alyssa |
| 5/5/2018 | Draft Edited | Alyssa |

1 Materials and Software Needs

This section will provide links on the things that we purchases, and the software needs to get an Arduino Sensor set-up.

1.1 Materials

Before we get down to the shopping list, there are a few essentials that you'll want to have. These include:

- A soldering station, including a soldering iron, cleaning materials, proper ventilation, and a pin-point solder iron tip
- A multimeter in the event you're unsure something is working the way it ought to.

The following items are items that we purchased to make our own sensor. This sensor measured only particulate matter, but you could purchase other sensors and follow similar instructions to achieve the same sort of results.

- Arduino Mega: https://www.amazon.com/gp/product/B0046AMGW0/ref=crt_ewc_title_gw_1?ie=UTF8&psc=1&smid=AM0JQO74J587C
- WiFi Breakout: <https://www.adafruit.com/product/2999#tutorials>
- PM Sensor: <https://www.adafruit.com/product/3686#tutorials>
- GPS Shield: <https://www.adafruit.com/product/1272#tutorials>
- GPS Antenna: <https://www.adafruit.com/product/960#tutorials>
- A printer USB cable (called a USB A to USB B cable): <https://www.adafruit.com/product/62>
- SMA to uFL Adapter to attach the GPS Antenna: <https://www.adafruit.com/product/851#tutorials>
- A hefty amount of jumper cables: <https://www.adafruit.com/product/153>
- Micro SD Card (not essential, but we did include one on our sensor)
- Breadboard: <https://www.adafruit.com/product/64#tutorials>
- Plastic Mounting Board to hold the Arduino and Breadboard: <https://www.adafruit.com/product/275#tutorials>
- Shield Stacking Headers: <https://www.adafruit.com/product/85#tutorials>

1.2 Software Needs

First and foremost, we recommend that you get a good text editor. There are many out there, we recommend Atom or Notepad++. You can download these programs from online.

Development for the Arduino is done on the Arduino's IDE. We used the most current version, version 1.8.5. For Windows, you can download that here: https://www.arduino.cc/download_handler.php?f=/arduino-1.8.5-windows.exe For Mac, you can download that here: https://www.arduino.cc/download_handler.php?f=/arduino-1.8.5-macosx.zip

Please note that the installation instructions that follow are designed for a Windows machine. If you have a Mac, they may differ slightly.

Run the file that you downloaded, there may be a security message that comes up and you can click "yes" on that. Next click "I agree." On this screen, make sure that all boxes are checked, and click "next." Lastly, click "install." When complete, simply click "close."

When the installation completes, you'll need to do some basic setup. First, go to "Tools" then "Board," and select Arduino Mega ADK. Another thing you'll need to take note of is the "Port." When you plug the Arduino in, you'll need to go to the port, and select the proper option. Usually, it will show something that says "COM# (Arduino/Genuino Mega or Mega 260)." If you get errors when uploading your code, verify that the proper port is selected.

Next we'll need to install a few libraries. To do this, go to "Sketch," then "Include Library," Then "Manage Libraries." From this window, type "wifi101" then hit enter. Click on the first one that comes up (the title should say "WiFi101" by "Arduino," we used version 0.15.2), and click "Install." Before you close this window, you'll need to install one more library. Search for "Adafruit_GPS" and install this library. We used version 1.0.3 for this.

Once you are done with this, it's time to begin building your node.

2 Building Your Node

The first thing you'll need to do is solder a header strip to the WiFi board. Never done this before? Don't fret, we had minimal soldering experience and it went well on our first try. Follow these links to get this step done:

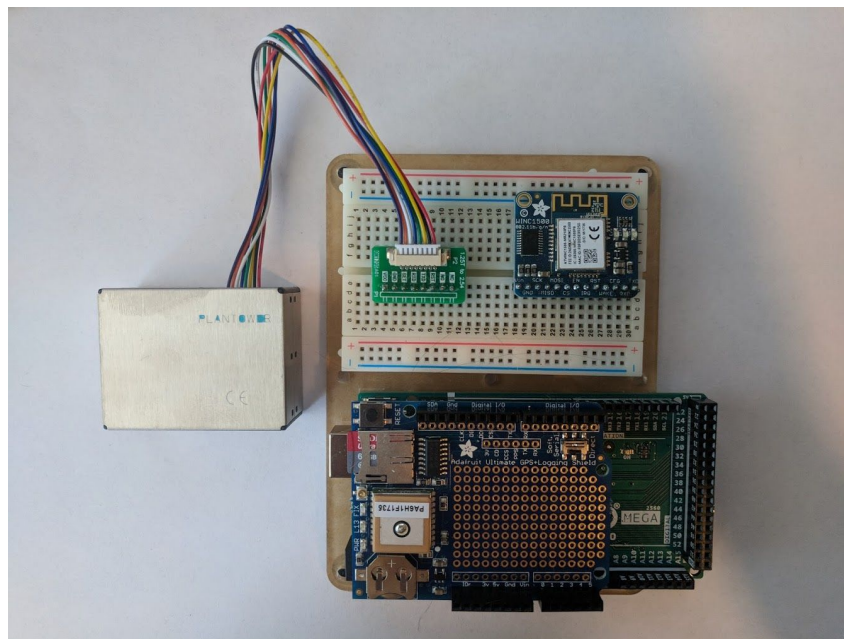
- Assembling the WiFi Breakout Board:
<https://learn.adafruit.com/adafruit-atwinc1500-wifi-module-breakout/assembly>
- How to Solder: <https://learn.adafruit.com/adafruit-guide-excellent-soldering>

Next, you'll need to solder some parts of the GPS board. Follow this link to accomplish this:
<https://learn.adafruit.com/adafruit-ultimate-gps-logger-shield/shield-headers>

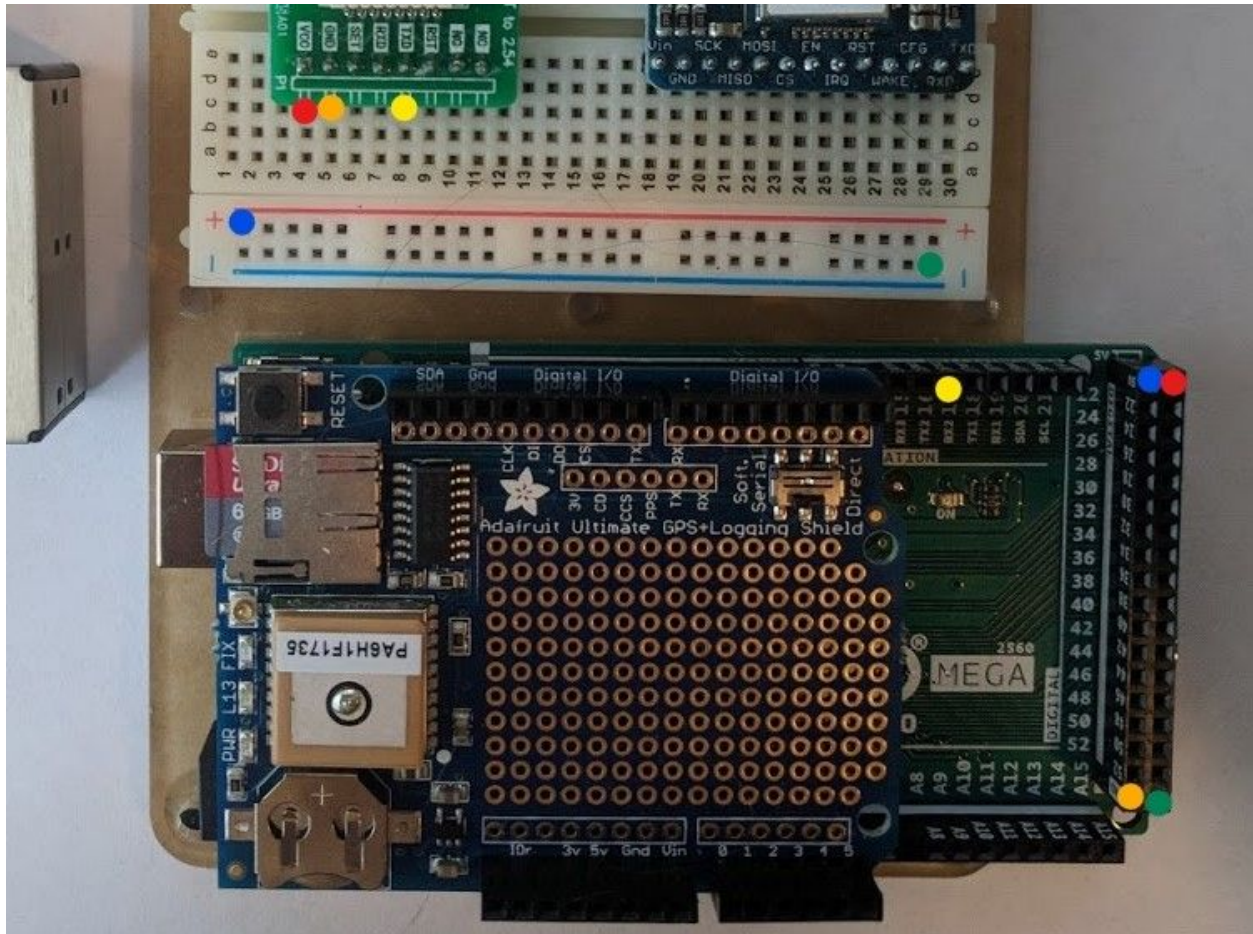
Lastly, you'll need to plug in the Particulate Matter Sensor (the box) to the breakout board using the included ribbon-wire.

With the soldering complete and wires connected, we can now begin setting up our Arduino. The first thing we recommend doing is assembling your mounting plate, if you chose to purchase one. It keeps the arduino in place and prevents it from sliding around.

First, you'll need to place the Particulate Matter Breakout and the Wifi Breakout boards to the breadboard. The Particulate Matter Board should go on the breadboard row titled "E," and the first pin (titled VCC) should go on column 4, the last pin (NC) should be on column 11. The WiFi board should also go along row "E," with the first pin on column 18 and the last on column 30. It should look something like this:



Next, we need to wire everything together, we'll take this one sensor at a time. First, let's start with the particulate matter sensor. Use the picture below as a guide to guide you on where to place the wires. Plug one end of the jumper cable to one color dot, and find the same color on the picture, and plug it into the appropriate place. A list is provided to guide you through any issues you might have following the picture.

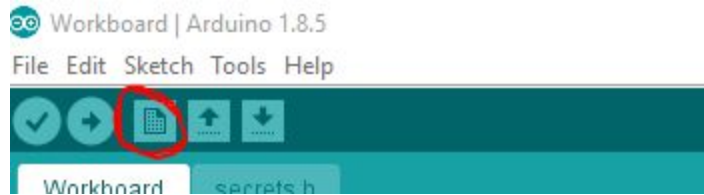


Wiring for the PM Sensor, 5 wires:

- Upper most Positive breadboard hole -> Arduino 5V (right above port 22)
- Bottom most Negative breadboard hole -> Arduino Ground (right below port 52)
- PM VCC (on the breadboard that would correlate to C, 4) -> Arduino 5V (right above port 23)
- PM GND (on the breadboard, that would correlate to C, 5) -> Arduino Ground (right below port 53)
- PM TXD (on the breadboard, that would correlate to C, 8) -> Arduino port number 17 (RX2).

This should be all you need to get the PM Sensor running. To confirm, we need to run some code on the Arduino.

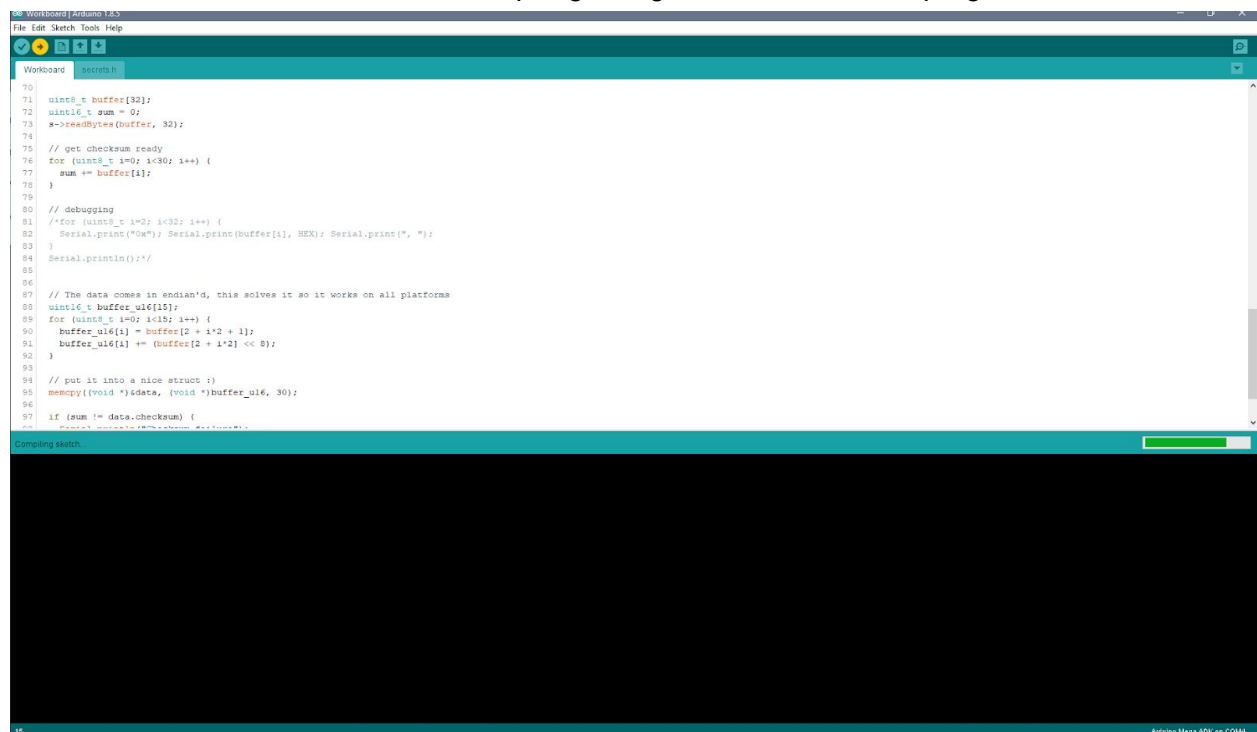
Plug in your Arduino to your computer using the USB A to USB B cable, and open up the Arduino IDE, and click the icon shown below to create a new file:



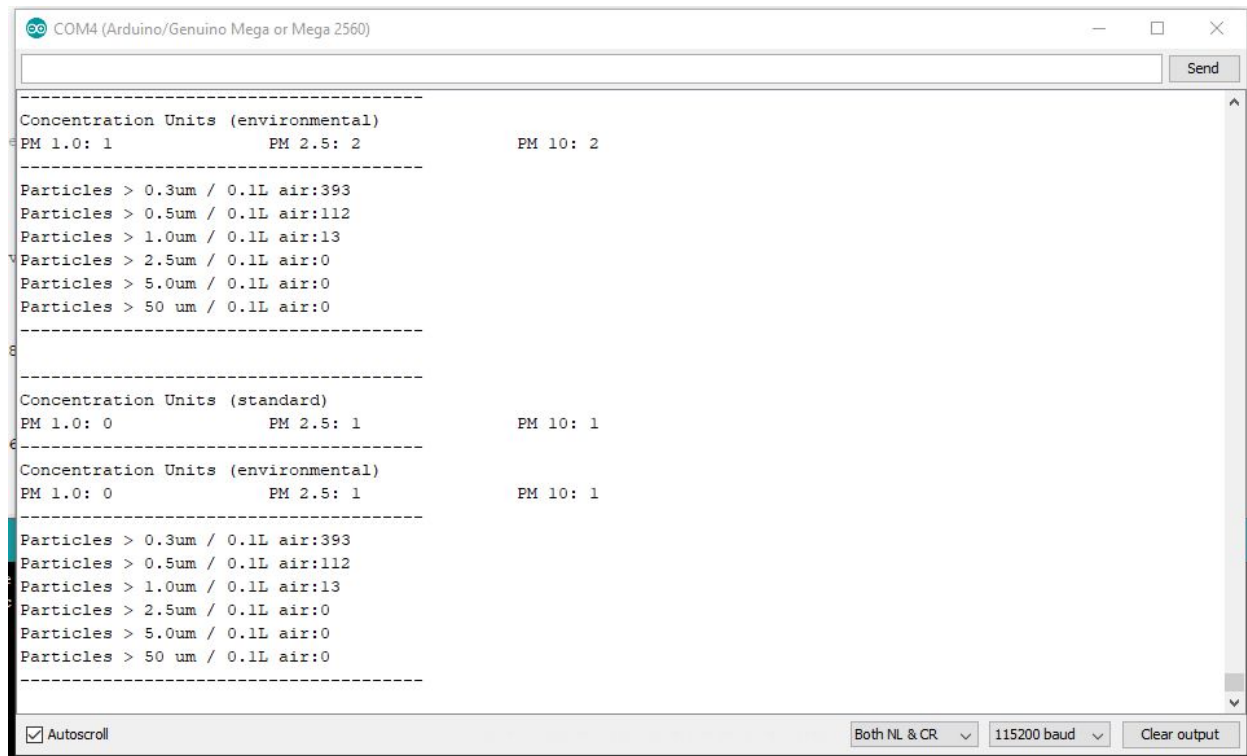
Once you create your new file, click “File” then “Save As” and name the file “Workboard” or something similar.

Copy and paste the code here to your Workboard file

(<https://github.com/Project-DAMN/Arduino-Code/blob/master/PM%20Sensor%20Test>), then click the arrow to the left of the “new file” icon to upload the file to the Arduino. This may take a few moments. You can monitor the progress from the blue bar on the lower part of the screen. Here, it indicates that the sketch is compiling, the green bar shows the progress.



When the uploading process is complete, go to tools, then select “Serial Monitor.” Near the bottom, you’ll see something that says “9600 baud,” change this to “115200 baud.” You should see output similar to the following image, which indicates that this sensor is working as expected.



The screenshot shows the Arduino IDE serial monitor window for a COM4 port. The window displays the output of a PM2.5 sensor. The data is organized into three sections, each separated by dashed lines. The first section shows 'Concentration Units (environmental)' with PM 1.0: 1, PM 2.5: 2, and PM 10: 2. The second section shows 'Concentration Units (standard)' with PM 1.0: 0, PM 2.5: 1, and PM 10: 1. The third section shows 'Concentration Units (environmental)' with PM 1.0: 0, PM 2.5: 1, and PM 10: 1. Each section also includes a list of particle counts for various size ranges: 0.3um, 0.5um, 1.0um, 2.5um, 5.0um, and 50 um. The particle counts for 0.3um, 0.5um, and 1.0um are 393, 112, and 13 respectively. The particle counts for 2.5um, 5.0um, and 50 um are 0. The window has a 'Send' button at the top right and a status bar at the bottom with 'Autoscroll' checked, 'Both NL & CR' selected, '115200 baud' selected, and a 'Clear output' button.

```
COM4 (Arduino/Genuino Mega or Mega 2560)

-----
Concentration Units (environmental)
PM 1.0: 1          PM 2.5: 2          PM 10: 2
-----
Particles > 0.3um / 0.1L air:393
Particles > 0.5um / 0.1L air:112
Particles > 1.0um / 0.1L air:13
Particles > 2.5um / 0.1L air:0
Particles > 5.0um / 0.1L air:0
Particles > 50 um / 0.1L air:0
-----

Concentration Units (standard)
PM 1.0: 0          PM 2.5: 1          PM 10: 1
-----

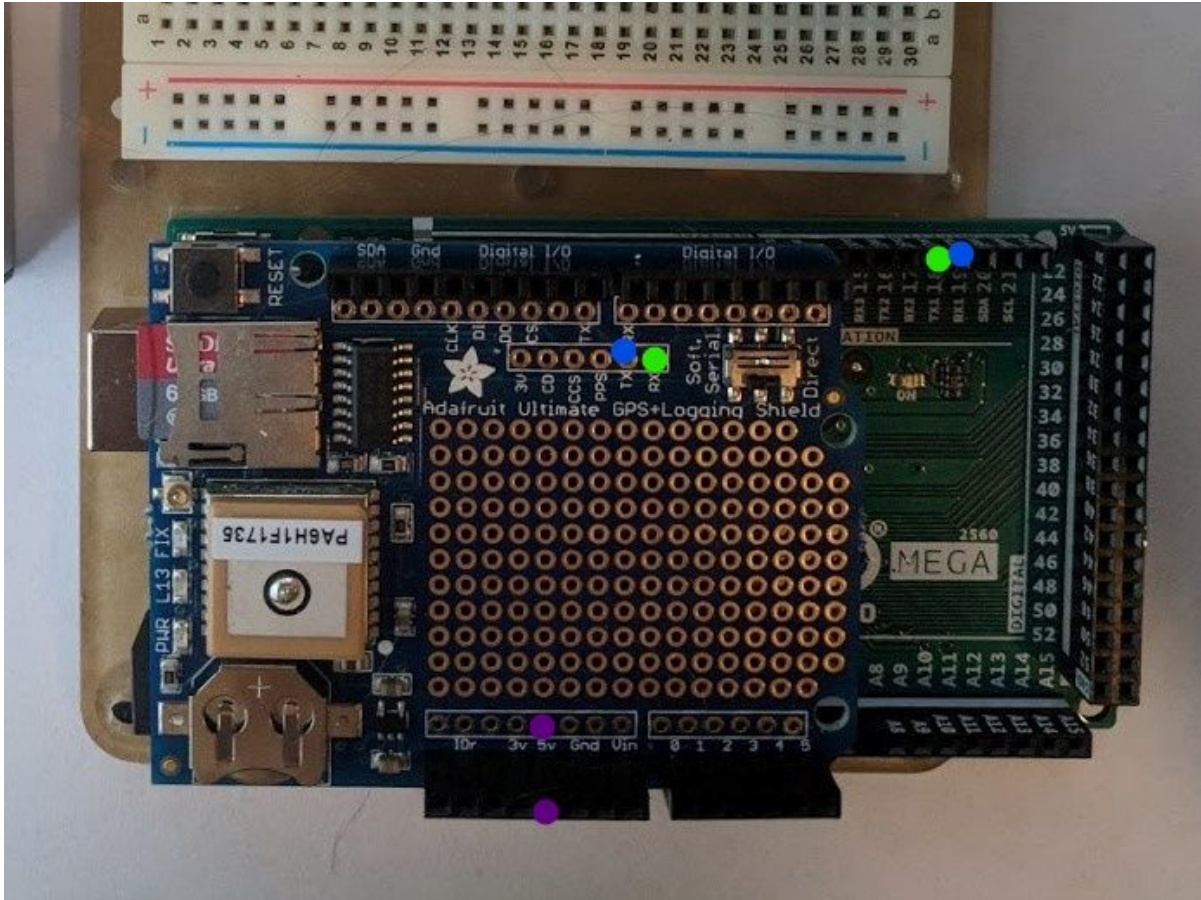
Concentration Units (environmental)
PM 1.0: 0          PM 2.5: 1          PM 10: 1
-----
Particles > 0.3um / 0.1L air:393
Particles > 0.5um / 0.1L air:112
Particles > 1.0um / 0.1L air:13
Particles > 2.5um / 0.1L air:0
Particles > 5.0um / 0.1L air:0
Particles > 50 um / 0.1L air:0
-----

[Autoscroll] [Both NL & CR] [115200 baud] [Clear output]
```

If you do not see numbers with the output, or if you see nothing at all, reread the instructions and make sure nothing was missed. For further help, see this website:

<https://learn.adafruit.com/pm25-air-quality-sensor/arduino-code>

Continuing with our wiring, the next thing we will test is our GPS sensor. First unplug the Arduino from USB, then wire up the following, in addition to what was wired in the previous step, do not unplug anything:



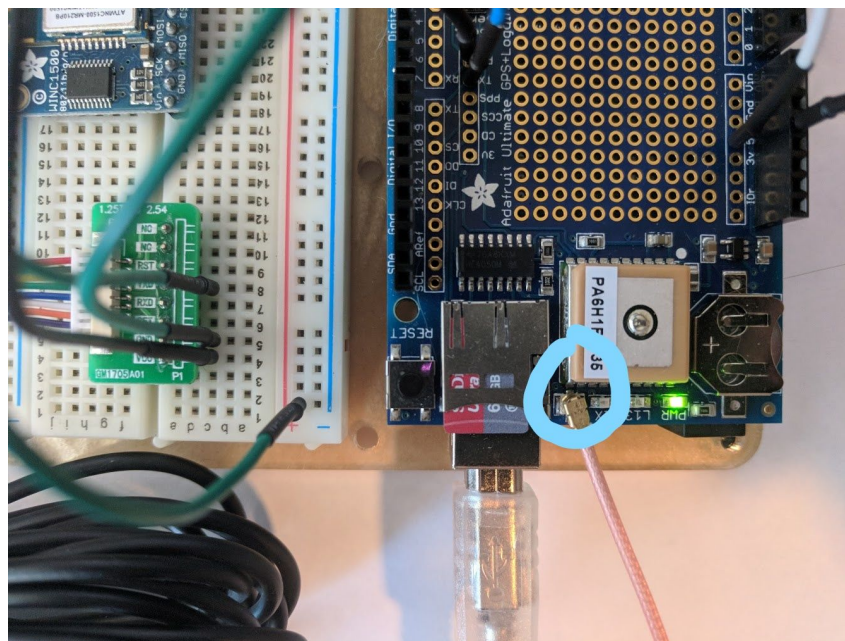
Wiring for the GPS Sensor, 3 Wires:

- 5V on the GPS Hat -> 5V on the Arduino. If you look at the side, underneath the GPS hat, you should be able to follow the pins to find the right hole. Otherwise, it should be the 5th hole from the left, if the unit is positioned the same way as shown on the above image.
- TX on the GPS Hat -> Arduino Port 19 (RX1). This can be confusing, but it is the lower most TX Port if you're looking at the arduino as it is oriented in the above image.
- RX on the GPS Hat -> Arduino Port 18 (TX1). This can be confusing, but it is the lower most TX Port if you're looking at the arduino as it is oriented in the above image.

Next, we should test the GPS Sensor. To do this, first attach the SMA to uFL Adapter to the GPS Antenna. It will look something like this.

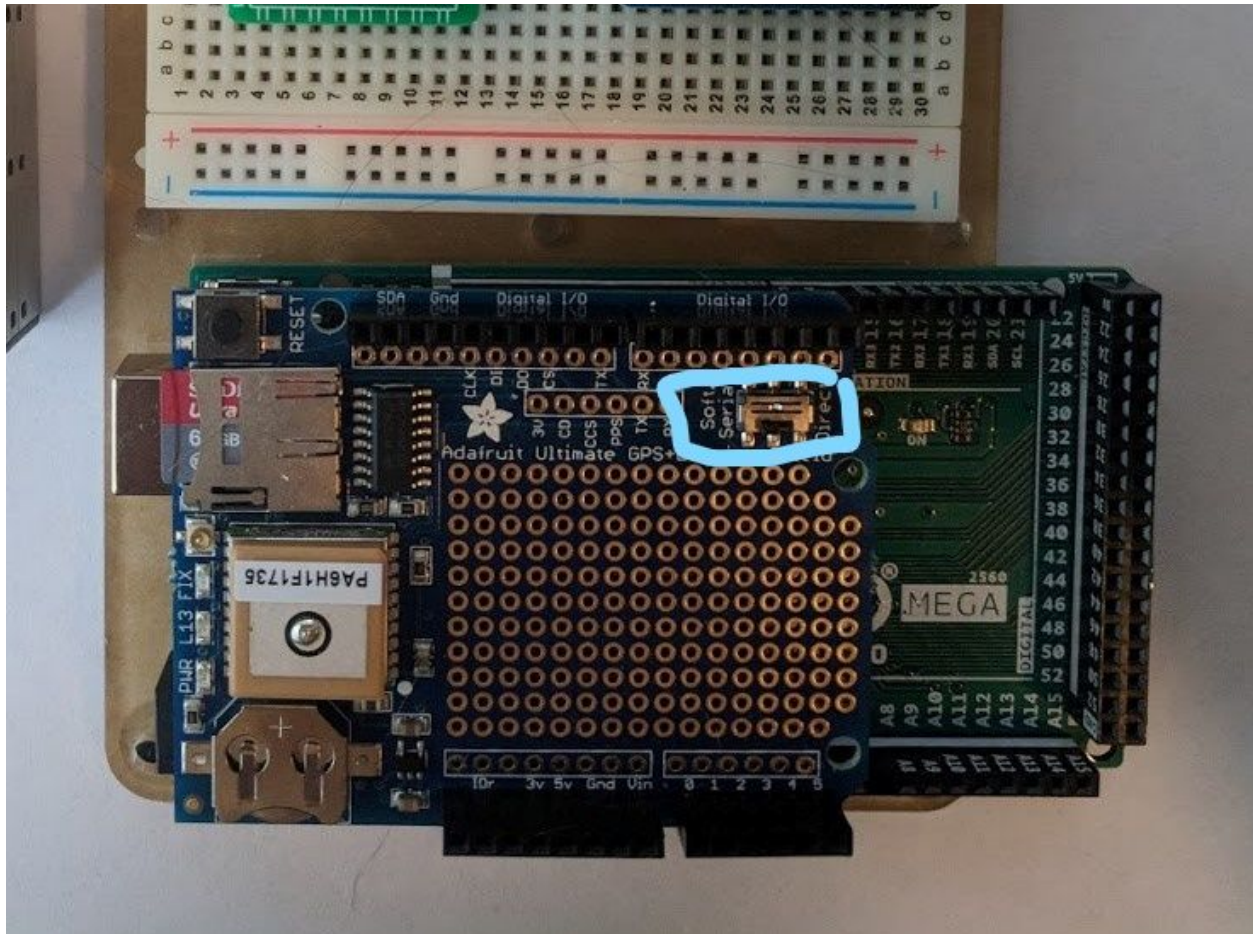


Then, attach the GPS Antenna to the GPS Shield. To do this, simply snap the end of the adapter onto the shield, it will look something like this:



Next, copy the code from here

(<https://github.com/Project-DAMN/Arduino-Code/blob/master/GPS%20Sensor%20Test1>) and replace with the code that we sent to the Arduino in the previous step. There is a switch on the GPS Sensor where it says soft serial, and direct. Flip the switch to direct.



Upload the code. If you don't see anything at first, make sure that you switch the baud from 115200 to 9600. The code should look something like this:

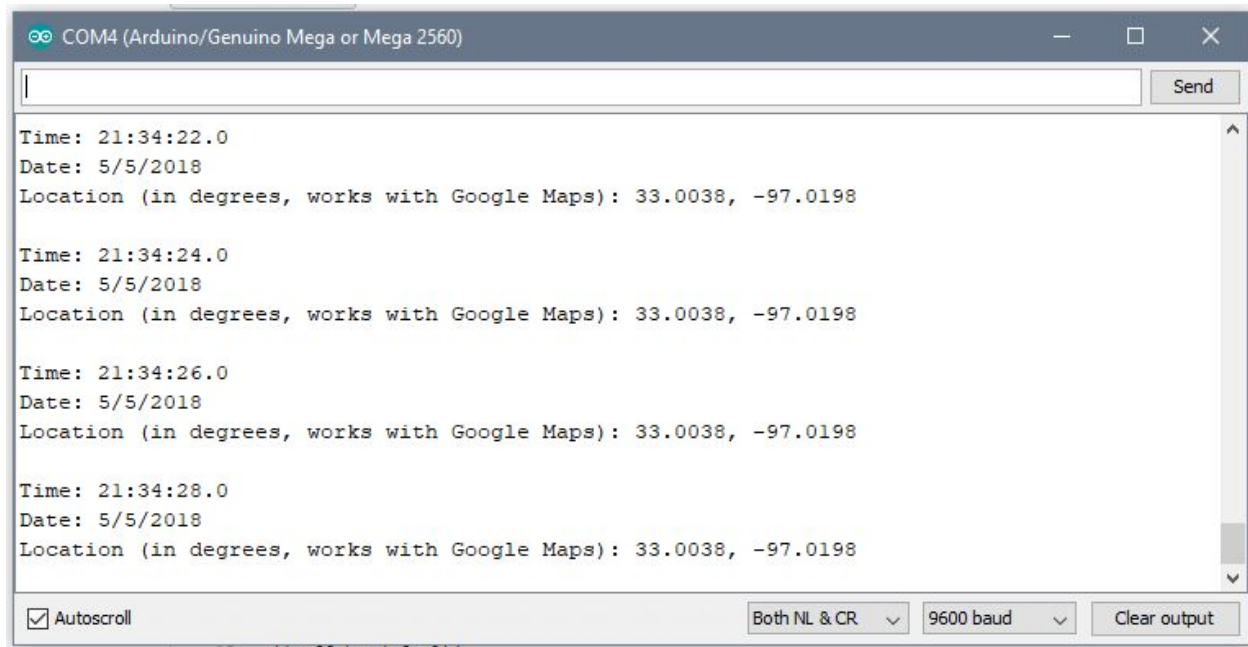
```
COM3 (Arduino/Genuino Mega or Mega 2560)
$GPRMC,225227.000,A,3300.2270,N,09701.1876,W,0.10,253.00,050518,,,A*72
$GPRMC,225228.000,A,3300.2269,N,09701.1876,W,0.08,138.05,050518,,,A*77
$GPRMC,225229.000,A,3300.2269,N,09701.1876,W,0.11,155.35,050518,,,A*76
$GPRMC,225230.000,A,3300.2269,N,09701.1876,W,0.17,170.72,050518,,,A*7C
$GPRMC,225231.000,A,3300.2269,N,09701.1876,W,0.16,167.47,050518,,,A*7C
$GPRMC,225232.000,A,3300.2268,N,09701.1876,W,0.17,194.87,050518,,,A*7F
```

As our next test, we're going to run some more code on the GPS Sensor. Flip the switch from direct back to soft serial, and copy the following code, replacing what was previously in the

arduino IDE, and upload this to your board:

<https://github.com/Project-DAMN/Arduino-Code/blob/master/GPS%20Sensor%20Test2>

This should produce output similar to the following:



The screenshot shows a serial monitor window titled "COM4 (Arduino/Genuino Mega or Mega 2560)". The window contains a text input field at the top with a "Send" button. Below the input field, the serial output displays four identical lines of data, each consisting of three lines of text: "Time: 21:34:22.0", "Date: 5/5/2018", and "Location (in degrees, works with Google Maps): 33.0038, -97.0198". The output is repeated for times 22.0, 24.0, 26.0, and 28.0. At the bottom of the window, there is a checkbox for "Autoscroll" which is checked, and three buttons: "Both NL & CR" (with a dropdown arrow), "9600 baud" (with a dropdown arrow), and "Clear output".

```
Time: 21:34:22.0
Date: 5/5/2018
Location (in degrees, works with Google Maps): 33.0038, -97.0198

Time: 21:34:24.0
Date: 5/5/2018
Location (in degrees, works with Google Maps): 33.0038, -97.0198

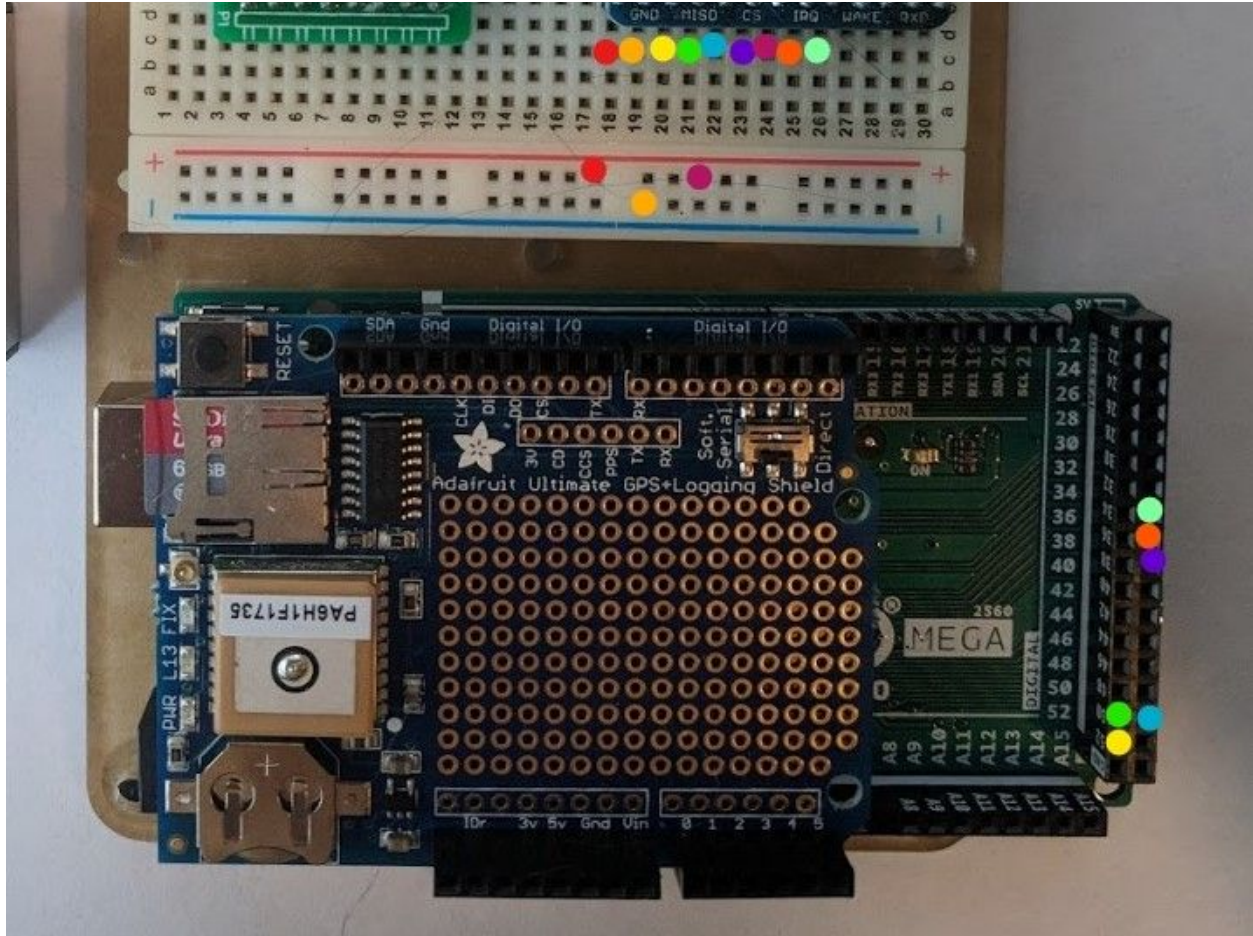
Time: 21:34:26.0
Date: 5/5/2018
Location (in degrees, works with Google Maps): 33.0038, -97.0198

Time: 21:34:28.0
Date: 5/5/2018
Location (in degrees, works with Google Maps): 33.0038, -97.0198
```

If you do not see numbers with the output, or if you see nothing at all, reread the instructions and make sure nothing was missed. For further help, see this website:

<https://learn.adafruit.com/adafruit-ultimate-gps-logger-shield/overview>

The last sensor that we need to connect is the WiFi Sensor. Same as before, don't disconnect anything, I'll show a picture of everything, and list out the connections following that.



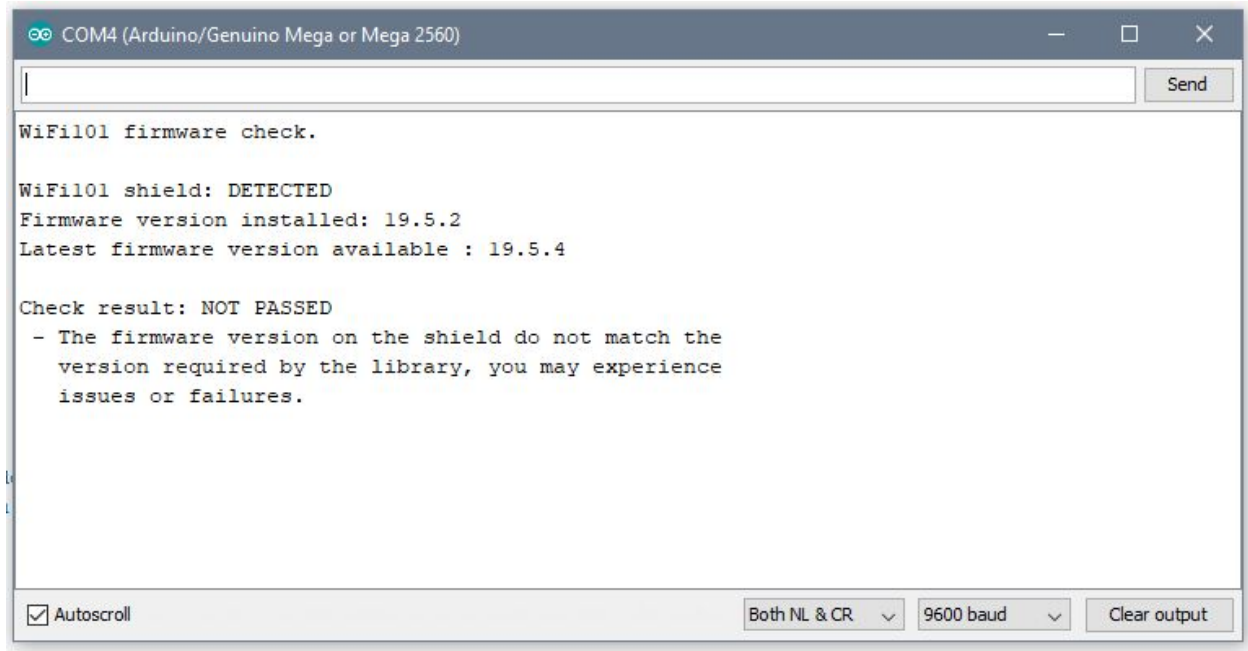
Wiring for the WiFi Breakout Board, 9 wires:

- Vin (C, 18 on the Breadboard) -> Positive area on the breadboard
- GND (C, 19 on the Breadboard) -> Negative area on the breadboard
- SCK (C, 20 on the Breadboard) -> Port 52 on the Arduino
- MISO (C, 21 on the Breadboard) -> Port 50 on the Arduino
- MOSI (C, 22 on the Breadboard) -> Port 51 on the Arduino
- CS (C, 23 on the Breadboard) -> Port 39 on the Arduino
- EN (C, 24 on the Breadboard) -> Positive area on the breadboard
- IRQ (C, 25 on the Breadboard) -> Port 37 on the Arduino
- RST (C, 26 on the Breadboard) -> Port 35 on the Arduino

Our test code for this shield can be copied from here:

<https://github.com/Project-DAMN/Arduino-Code/blob/master/WiFi%20Test>

Copy the code from there, plug your Arduino in, replace the code that was in the IDE with the code you copied, upload it, and verify that the output in your Serial Monitor looks something similar to the following, if it does, keep moving along. Ignore the message about "NOT PASSING":



If you do not see this output, or if you see nothing at all, reread the instructions and make sure nothing was missed. For further help, see this website, noting that the type of Arduino and breadboard used are different:

<https://learn.adafruit.com/adafruit-atwinc1500-wifi-module-breakout/wiring-and-test>

Now that you have your Arduino assembled and you have tested each sensor to verify functionality, it is time to upload the final code to the Arduino and run it. First, copy and replace the code in the IDE with the code found here, then save your file. Don't run it just yet, or you'll get some scary looking errors. Just save the file, then close out of the Arduino IDE for now.

<https://github.com/Project-DAMN/Arduino-Code/blob/master/Final%20Code>

Before we upload the file, there is one last thing we need to do. Near the top of the Arduino IDE, you'll see a little triangle pointing down. Click that, and then click "new tab." Type the name "secrets.h" and click OK. Lastly, add the following text to the file:

```
1 #define SSID ""
2 #define PASS ""
```

Enter in the name of your WiFi network between the quotes on line one, and the password to your network on line two.

First open "My Computer" (or more formally known as "File Explorer"). You can also do this by hitting the Windows Key and Letter E at the same time on your keyboard. Then type C:\Program Files (x86)\Arduino\hardware\arduino\avr\libraries\SoftwareSerial\srcv into the Address bar.

Right click "SoftwareSerial.cpp", and click edit, either with Notepad or Notepad++. For those on a Mac, you'll need to navigate to this same file.

Next, you'll need to change a few lines of code. Add `"/"` to line number 227, and `"*/"` to line 244, like so:

```
206  }
207  }
208
209  uint8_t SoftwareSerial::rx_pin_read()
210  {
211      return *_receivePortRegister & _receiveBitMask;
212  }
213
214  //
215  // Interrupt handling
216  //
217
218  /* static */
219  inline void SoftwareSerial::handle_interrupt()
220  {
221      if (active_object)
222      {
223          active_object->recv();
224      }
225  }
226
227  /*#if defined(PCINT0_vect)
228  ISR(PCINT0_vect)
229  {
230      SoftwareSerial::handle_interrupt();
231  }
232  #endif
233
234  #if defined(PCINT1_vect)
235  ISR(PCINT1_vect, ISR_ALIASOF(PCINT0_vect));
236  #endif
237
238  #if defined(PCINT2_vect)
239  ISR(PCINT2_vect, ISR_ALIASOF(PCINT0_vect));
240  #endif
241
242  #if defined(PCINT3_vect)
243  ISR(PCINT3_vect, ISR_ALIASOF(PCINT0_vect));
244  #endif*/
245
246  //
247  // Constructor
248  //
249  SoftwareSerial::SoftwareSerial(uint8_t receivePin, uint8_t transmitPin, bool inverse_logic /* = false */) :
250      _rx_delay_centering(0),
251      _rx_delay_intrabit(0),
252      _rx_delay_stopbit(0),
253      _tx_delay(0),
254      _buffer_overflow(false),
255      _inverse_logic(inverse_logic)
256  {
257      setTX(transmitPin);
258      setRX(receivePin);
259  }
260
261  //
262  // Destructor
263  //
264  SoftwareSerial::~SoftwareSerial()
265  {
266      end();
267  }
268
269  void SoftwareSerial::setTX(uint8_t tx)
270  {
271      // First write, then set output. If we do this the other way around,
272      // the pin would be output low for a short while before switching to
273      // output high. Now, it is input with pullup for a short while, which
274      // is fine. With inverse logic, either order is fine.
```

Save the file when you are done adding the appropriate information.

Now, click back on the tab that says “Workboard,” or whatever you named the file, and upload it to your Arduino and see if the output looks similar to this:

```
Connected.
Starting server connection.
Connected.

Concentration Units (environmental)
PM 1.0: 1          PM 2.5: 1          PM 10: 1

Concentration Units (environmental)
PM 1.0: 1          PM 2.5: 1          PM 10: 2

Concentration Units (environmental)
PM 1.0: 1          PM 2.5: 1          PM 10: 2

Concentration Units (environmental)
PM 1.0: 1          PM 2.5: 1          PM 10: 2

Concentration Units (environmental)
PM 1.0: 1          PM 2.5: 1          PM 10: 2

Concentration Units (environmental)
PM 1.0: 1          PM 2.5: 1          PM 10: 1

Time: 22:13:44.0
Date: 5/5/18
Location (in degrees, works with Google Maps): 33.0039, -97.0198
```

If you're not seeing anything pull up, first be patient. In order for anything to show up, the node has to make a connection with your WiFi, if after about a minute or so, you still don't see anything, verify that your WiFi connection information in Secrets.h is correct.

If it does, congrats! You have a working node.

3 Troubleshooting

This section will go through some basic troubleshooting that may be experienced with the nodes.

3.1.0 Code Won't Upload

First, go to “Tools” then “Board,” and select Arduino Mega ADK. Another thing you'll need to take note of is the “Port.” When you plug the Arduino in, you'll need to go to the port, and select the proper option. Usually, it will show something that says “COM# (Arduino/Genuino Mega or Mega 260).” If you get errors when uploading your code, verify that the proper port is selected.

If the code still isn't uploading, press upload again. In this instance, it's okay to be a little impatient.

3.1.1 Nothing Shows Up on the Serial Monitor

This is usually caused by wiring issues or the baud rate not being set right. Verify both of these items, and also try unplugging the Arduino, and plugging it back in.