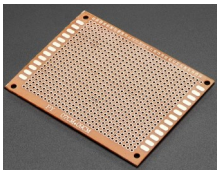
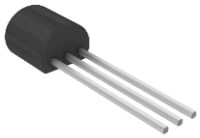
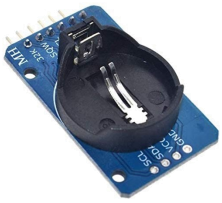
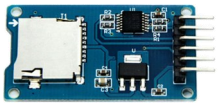
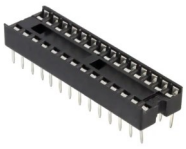
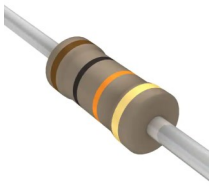
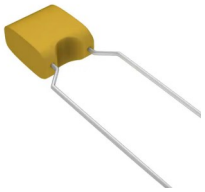
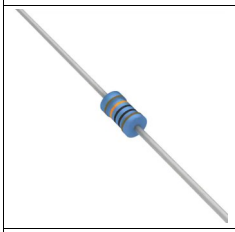
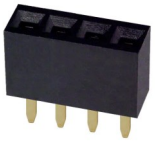
Wave Gauge Assembly and Operation Guide

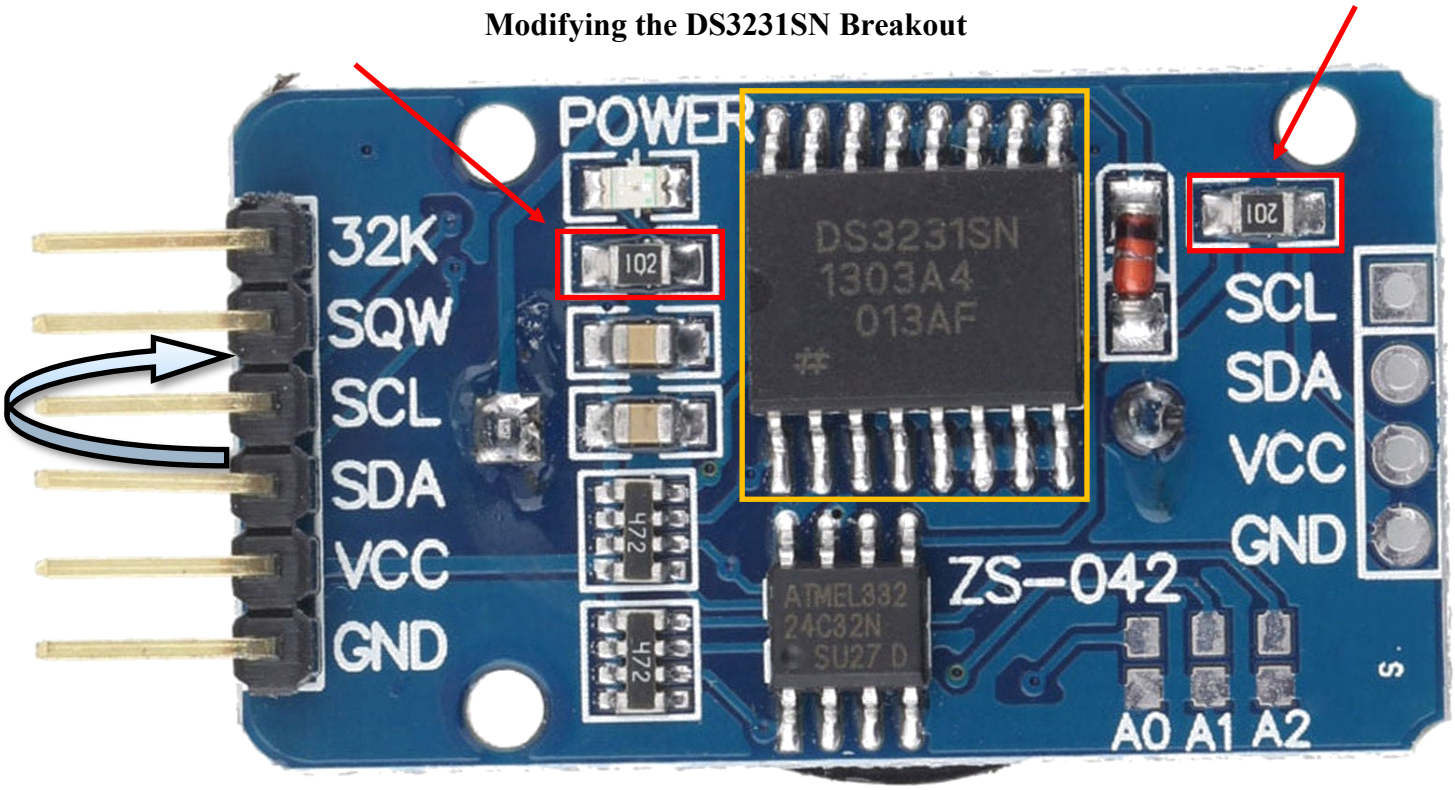
Circuit Board Component List

Image	Part	Quantity	Image	Part	Quantity
	Perfboard	1		3V Linear Voltage Regulator	1
	DS3231SN Breakout	1		Red LED	1
	MicroSD Breakout	1		2 Pin Male JST Connector	1
	28 Pin Dip Socket	1		1 µF Capacitor	1
	10 KΩ Resistor	2		0.1 µF Capacitor	3
	130 Ω Resistor	1		Female 3 Pin Header	1
	MS5803-14BA	1		Female 4 Pin Header	1
	SMT Breakout PCB for SOIC-8, MSOP-8 or TSSOP-8	1		(Preferably Blue) Lengths of Wire	1 x ~1 in. 1 x ~1.5 in.

Assembly Notes

- Practice soldering on an empty breadboard before starting this build if you aren't very comfortable with it. These are the essential skills for putting together the board:
    - Attempt to make lines of solder next to each other without shorting them together. One method of doing this is to first fill in the holes you wish to be connected using solder, without worrying about connecting them together. Then, place your soldering iron over each pair of adjacent holes to be connected and apply additional solder until they bridge together. Remove your soldering iron in a quick, straight up motion for best results.
    - When a short is inevitably created between two lines that aren't supposed to be connected, also practice fixing the connection: either wait for it to cool and then apply heat only to one hole (melted solder is attracted to heat), or use solder wick to remove solder from the board.
    - Finally, practice soldering components' leads / pins or wire to the board. Desoldering components with several leads can be very difficult with just a soldering iron, so it's important to place them in the right holes the first time. To solder a lead in place, rest the tip of the soldering iron against both the pin of the component and the copper ring on the board. Then, simply apply solder to the ring, close to your iron, until the entire hole is filled.
- The board can be put together in *almost** any order, but the steps below are organized for ease of soldering.
  * Two jumper wires are used in the build: these wires will be covered when the RTC and SD breakout boards are placed on the board, so they must be at least placed in the proper holes before soldering the RTC and microSD breakouts.
- All components and jumper wires are placed on the front of the perfboard, and soldering is done on the back. The back can be identified by the copper rings around each of the holes.

Modifying the DS3231SN Breakout



The two resistors (red boxes) should be removed from the module. Do this by applying heat using a soldering iron to one pad of the resistor and pushing until it moves off the pad, or by repeatedly applying heat to both pads and pushing. We won't be using these resistors later, so it's alright if they are damaged.

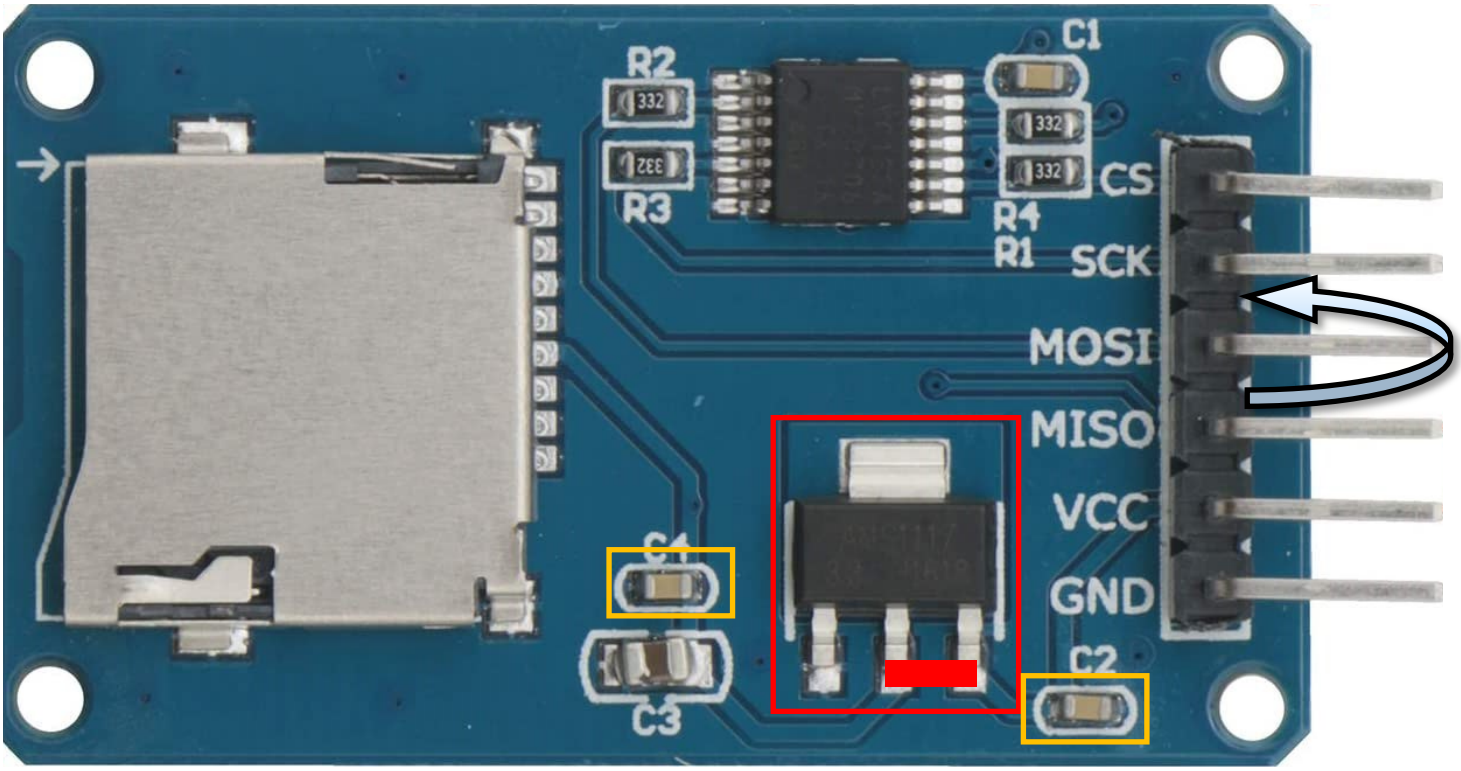
- We remove the left resistor to open the circuit to the power LED above it, which is unnecessary and wastes battery power in our setup. The power LED can also be optionally removed, but since the circuit is already open it makes no difference to power consumption.
- The right resistor is used to allow a small amount of current from VCC to trickle charge the coin cell. We remove it to open this battery charging circuit, as we are using non-rechargeable batteries which would be damaged by it. Even if you use rechargeable batteries, our VCC voltage of 3V may not be high enough to charge them properly. The red diode to the left of this resistor can also be optionally removed.

After removing the components, bend the 6 pins on the left so they come straight out of the board, rather than being parallel to it as shown in the picture.

It is also important to observe the type of DS3231 chip on the board (orange box). Notice this chip is labelled as "DS3231SN". This is the variant we want.

- The "SN" variant of the chip can be used as both a RTC (real time clock) and TCXO (temperature compensated crystal oscillator); both of these features are necessary for our sensor design. Other versions of the chip exist, all of which can be used as a RTC of course, but some lack the ability to be used as a TCXO and are less accurate time keepers, primarily the DS3231M variant. These variants cannot be used, and this seems to be one of the tradeoffs of buying these boards so cheaply: sellers will not always guarantee or even mention what variant of chip you will receive.

Modifying the MicroSD Breakout



First, the voltage regulator (red box) must be removed from the module. Since our design already includes a voltage regulator (3V), this one is unnecessary and wastes power.

- To remove it using a soldering iron, repeatedly apply heat and force to the 4 pins (3 on the bottom, 1 on the top) until it can be pushed loose. We won't be using this part later, so it's alright if it is damaged. Another way is to only apply heat to the pins on one side while using your iron (or another slim object) to leverage that side into the air by pushing up from beneath the leads.

Once removed, there is no longer a path for power to reach the card reader. To fix this, we will short the voltage regulator's input pad to it's output pad (the smaller filled in red rectangle).

- This is safe since our input is already at 3V, but using a higher voltage could damage the SD card. Place your soldering iron over both pads and apply solder until they bridge together (this may require some fiddling). Once bridged, remove your soldering iron straight upwards in a swift motion for the best chance of the pads staying together. Make sure that NO path is formed to the leftmost or top pads (GND), as this will create a short circuit.

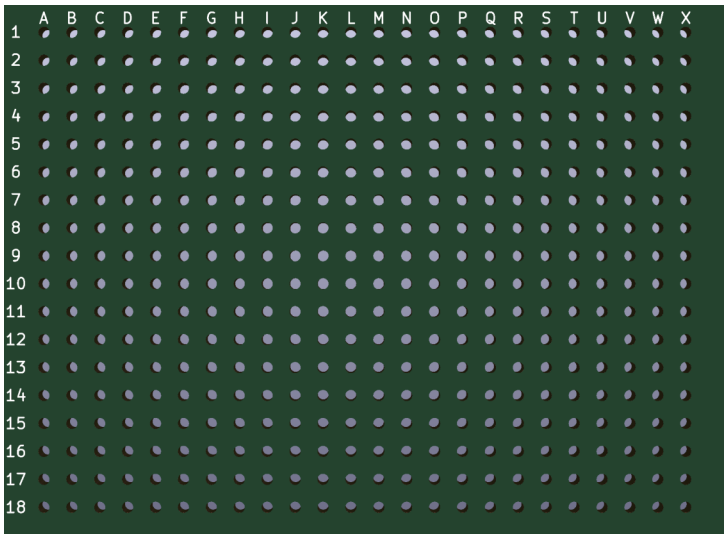
Next, we can remove capacitors C2 and C4, marked with the orange boxes. This isn't necessary for the board to function, but may save a small amount of power.

- These capacitors can be removed similarly to the resistors from the RTC module: either apply heat to both sides repeatedly or stick to one side and push until it levers upward. These components won't be used later.

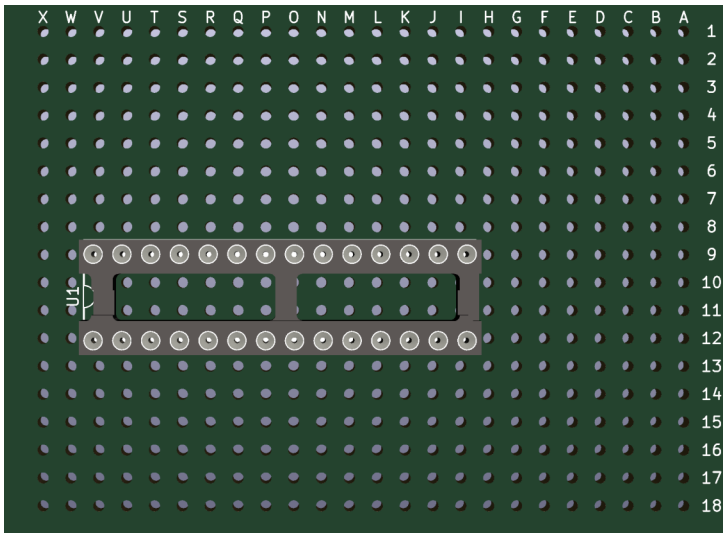
Finally, we again bend the 6 pins on the right so they are perpendicular to the board.

Board Assembly

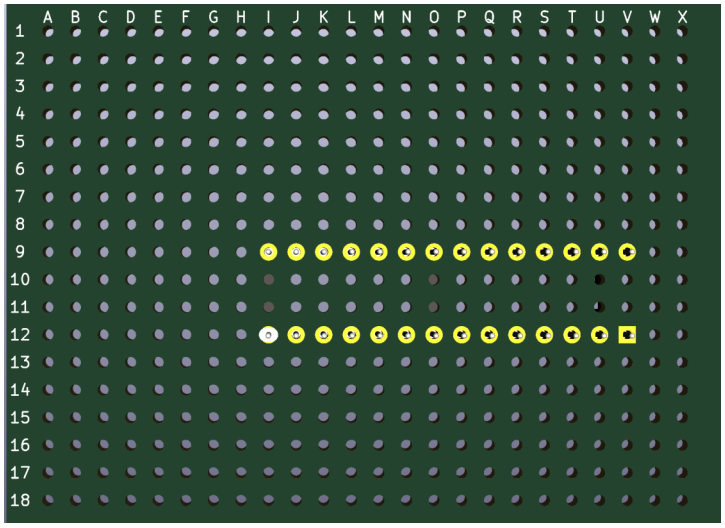
Step 1: Start with a blank piece of Perfboard.



Step 2: Place the 28 Pin DIP Socket through holes V9, V12, K12, K9. The orientation of this piece does not matter.

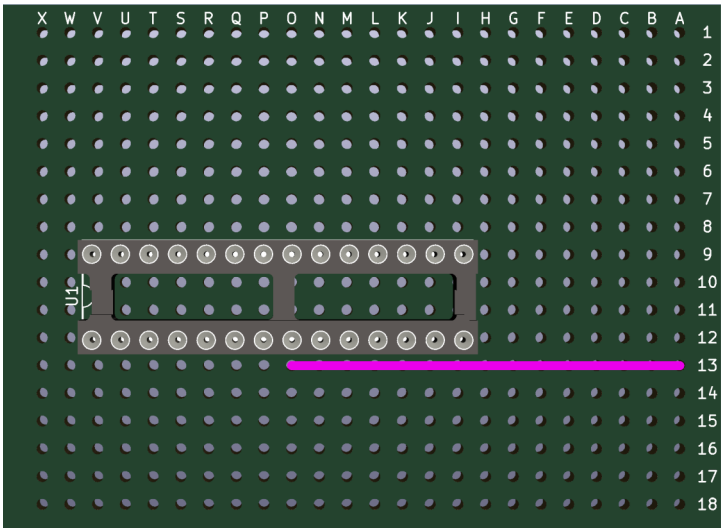


Step 3: Solder pin I12 to hold the socket in place.

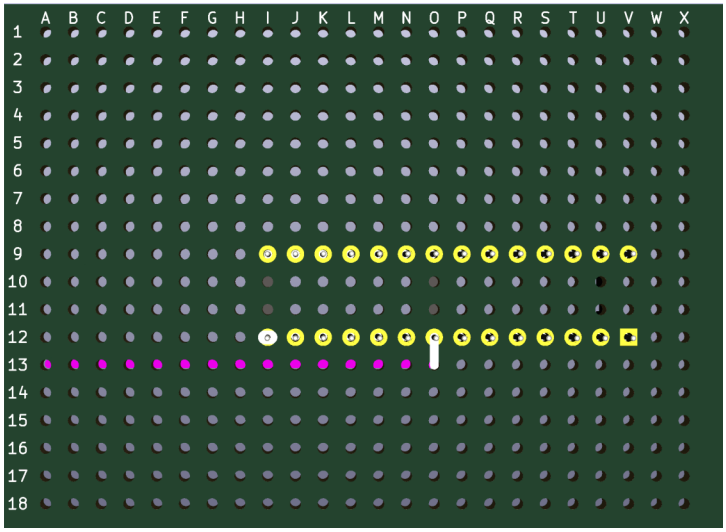


Board Assembly

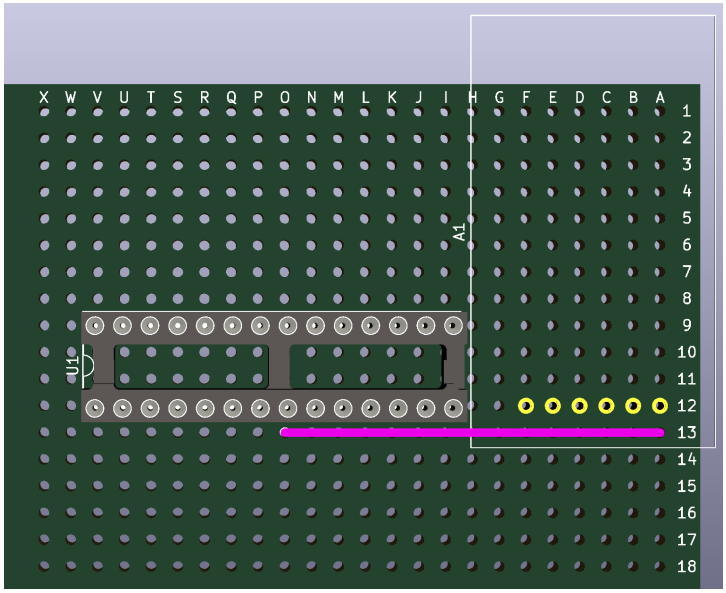
Step 4: Place a jumper cable between O13 and A13. Make sure the cable is on the front of the perfboard, and that the wires are long enough to poke through each hole.



Step 5: Solder connections:
O12 to O13

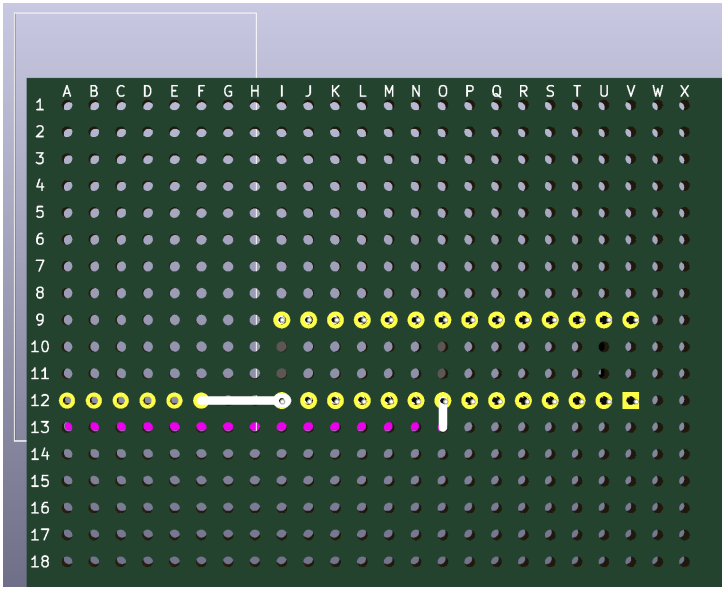


Step 6: Place the MicroSD Breakout onto the perfboard:
GND goes through A12
VCC goes through B12
MISO goes through C12
MOSI goes through D12
SCK goes through E12
CS goes through F12

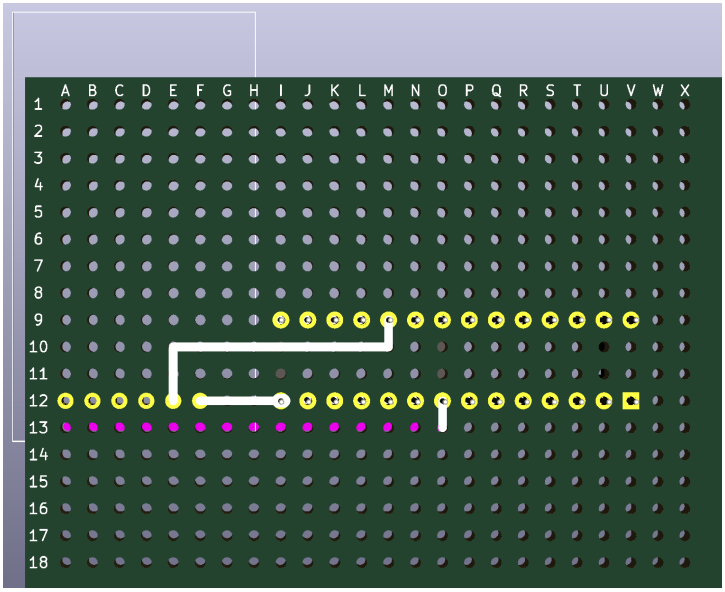


Board Assembly

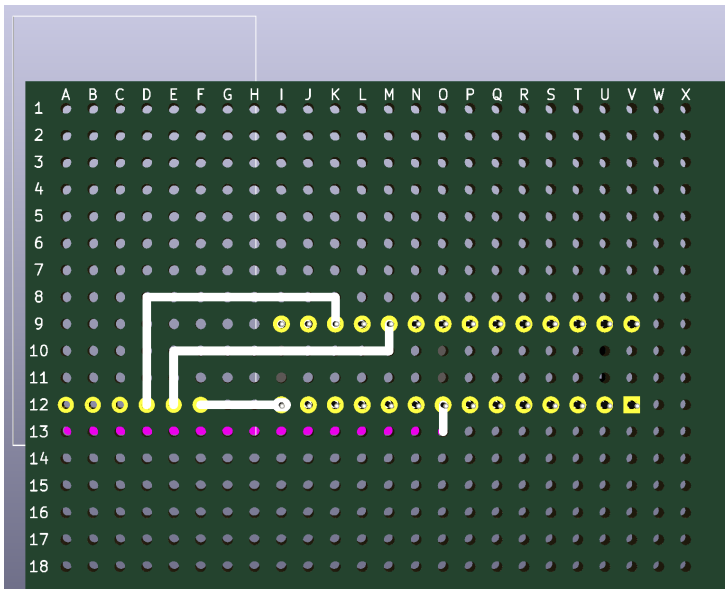
Step 7: Solder connections:
F12 to I12



Step 8: Solder connections:
E12 to E10 to M10 to M9

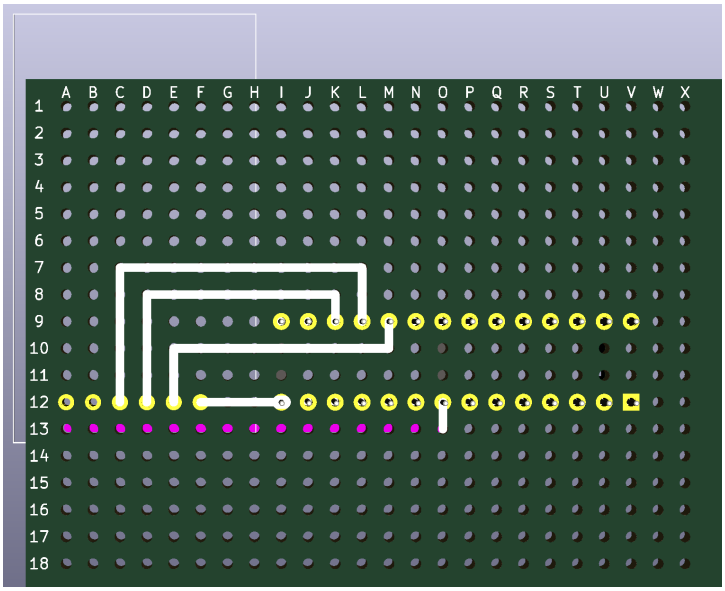


Step 9: Solder connections:
D12 to D8 to K8 to K9

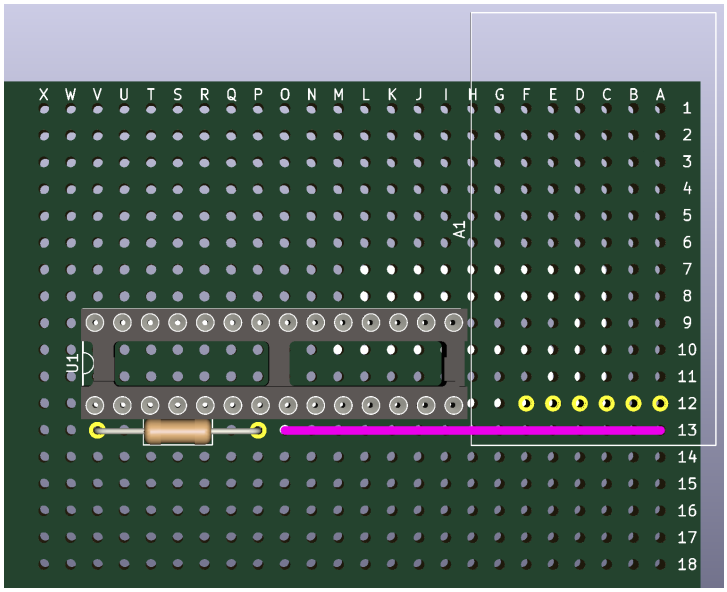


Board Assembly

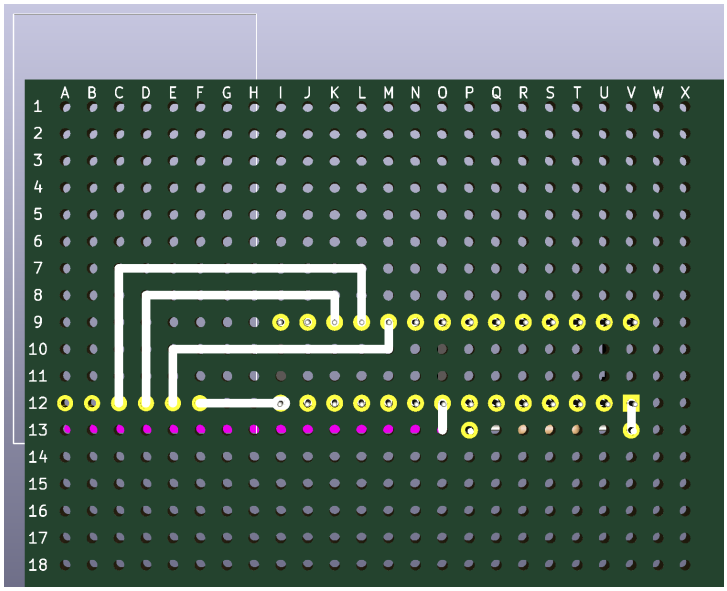
Step 10: Solder connections:
C12 to C7 to L7 to L9



Step 11: Place a 10 KΩ Resistor between V13 and P13. The orientation of this piece does not matter.

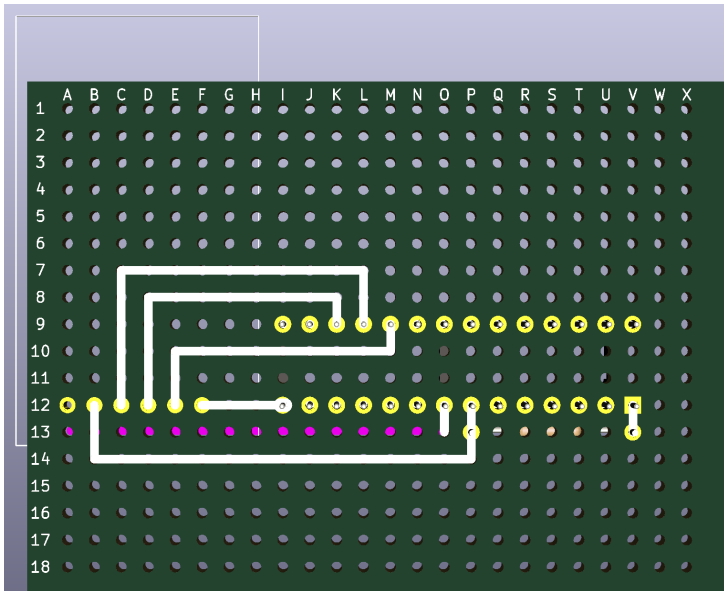


Step 12: Solder connections:
V12 to V13

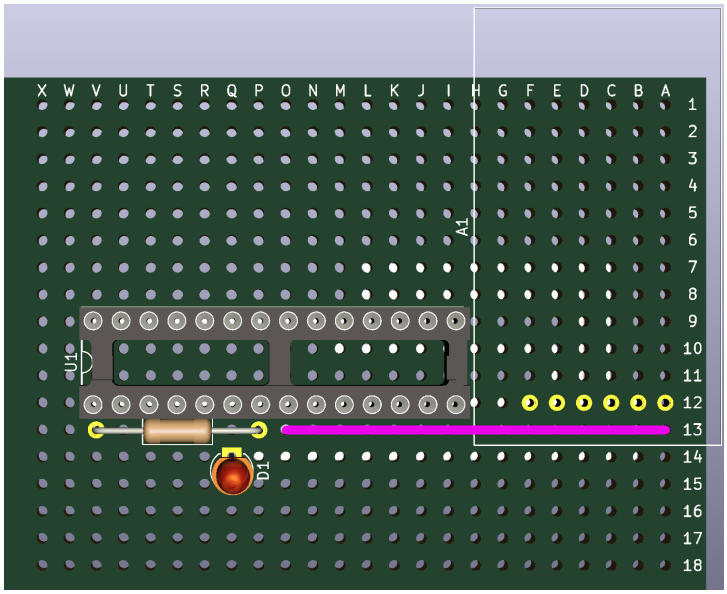


Board Assembly

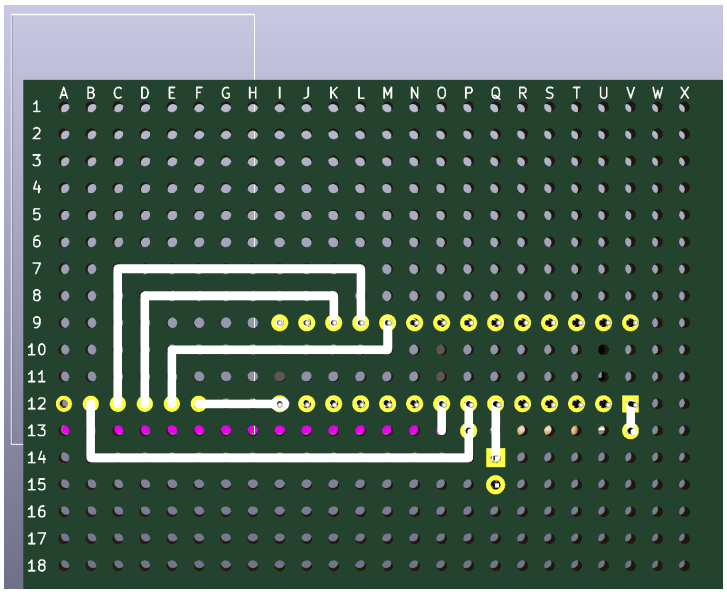
Step 13: Solder connections:
B12 to B14 to P14 to P12



Step 14: Place the Red LED on the board, with the longer lead going through Q14 and the shorter lead going through Q15. The orientation of this piece DOES matter.

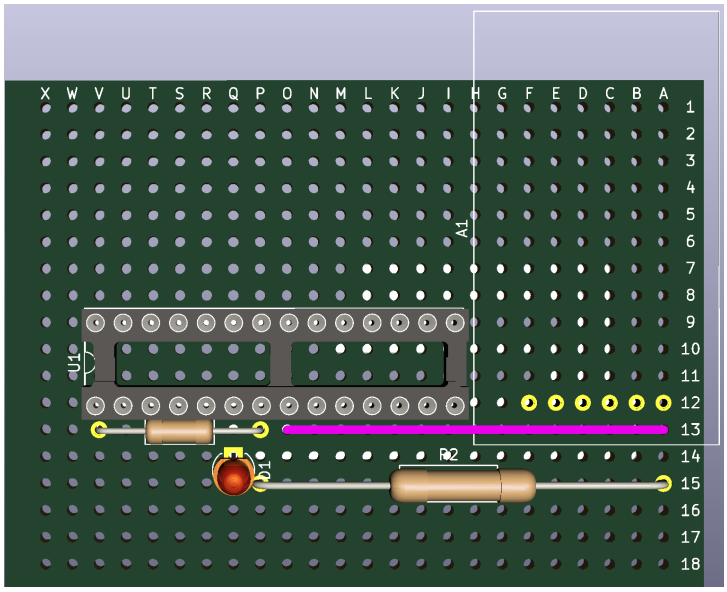


Step 15: Solder connections:
Q12 to Q14

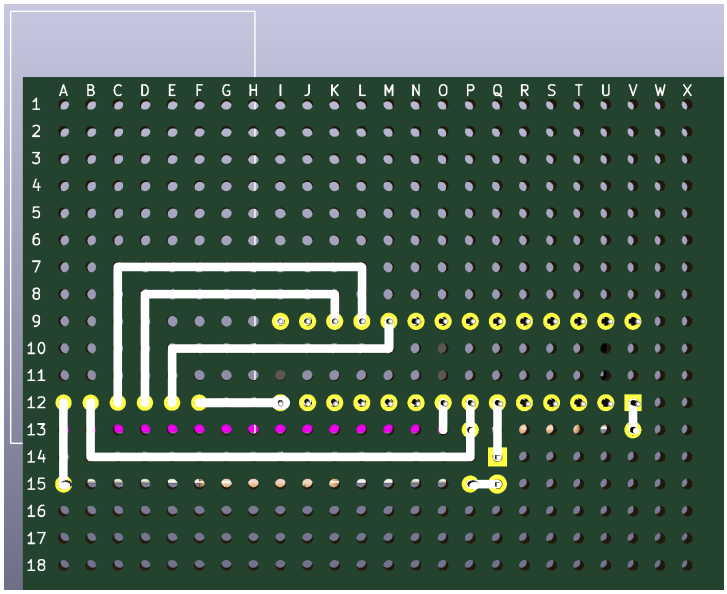


Board Assembly

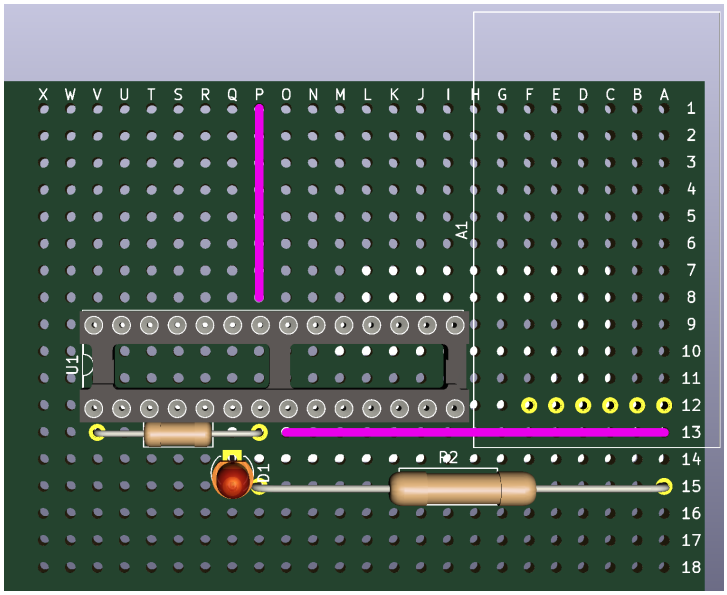
Step 16: Place the 130 Ω Resistor across pins P15 and A15. The orientation of this piece does not matter.



Step 17: Solder connections:
P15 to Q15
A12 to A15

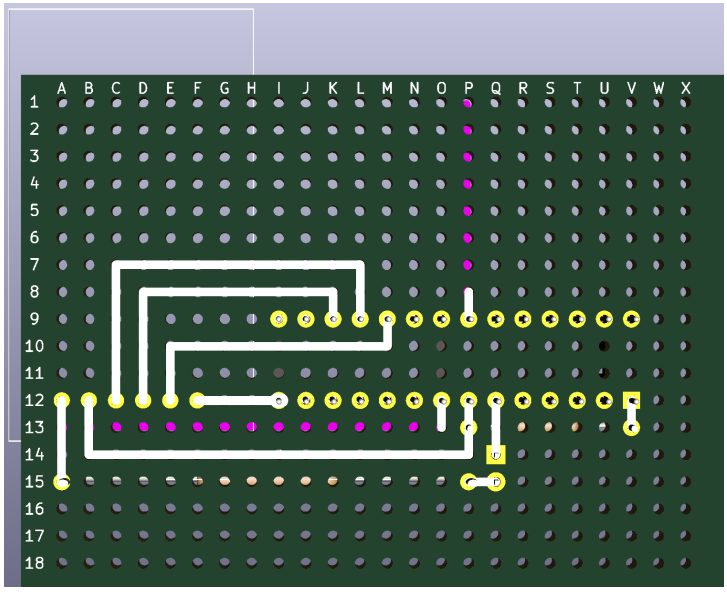


Step 18: Place a jumper cable between P1 and P8. Make sure the cable is on the front of the perfboard, and that the wires are long enough to poke through each hole.

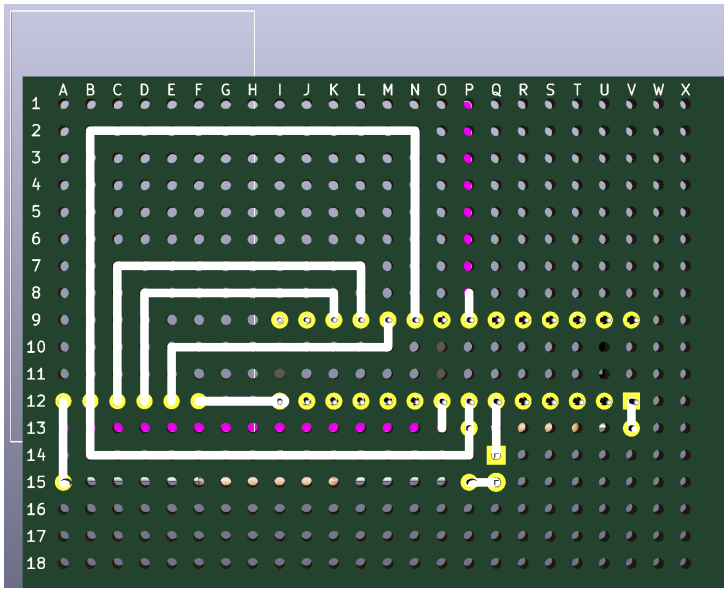


Board Assembly

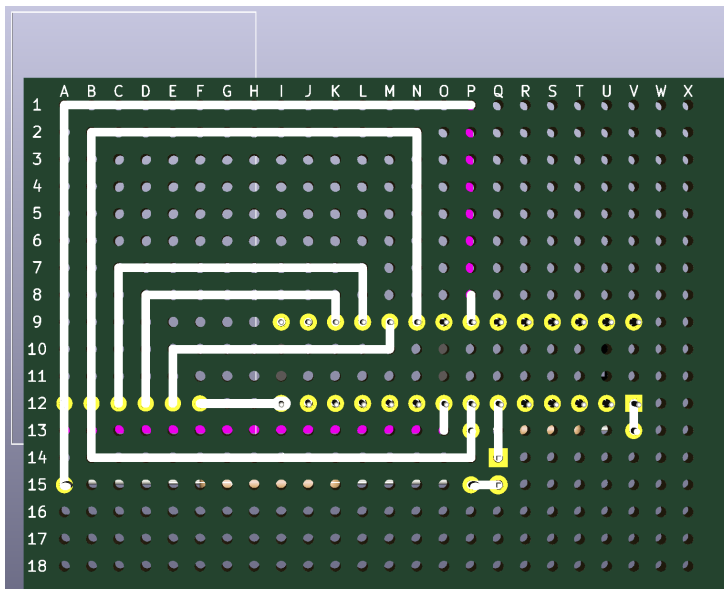
Step 19: Solder connections:
P8 to P9



Step 20: Solder connections:
B12 to B2 to N2 to N9

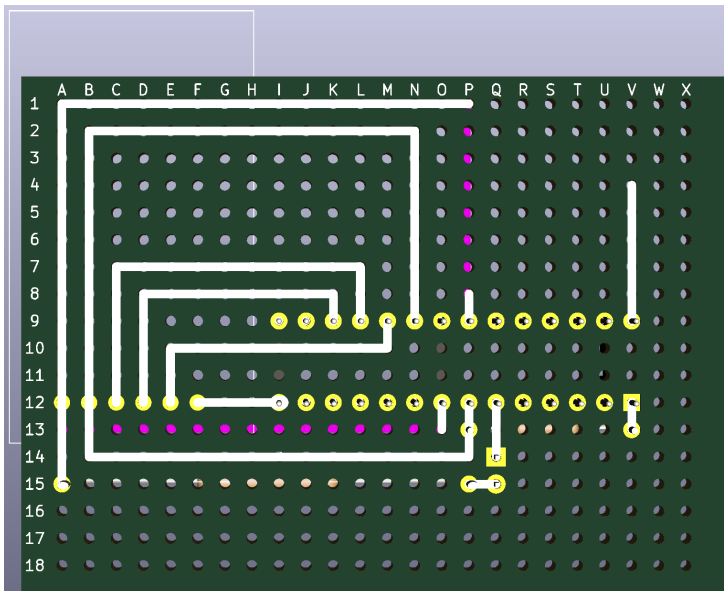


Step 21: Solder connections:
A12 to A1 to P1

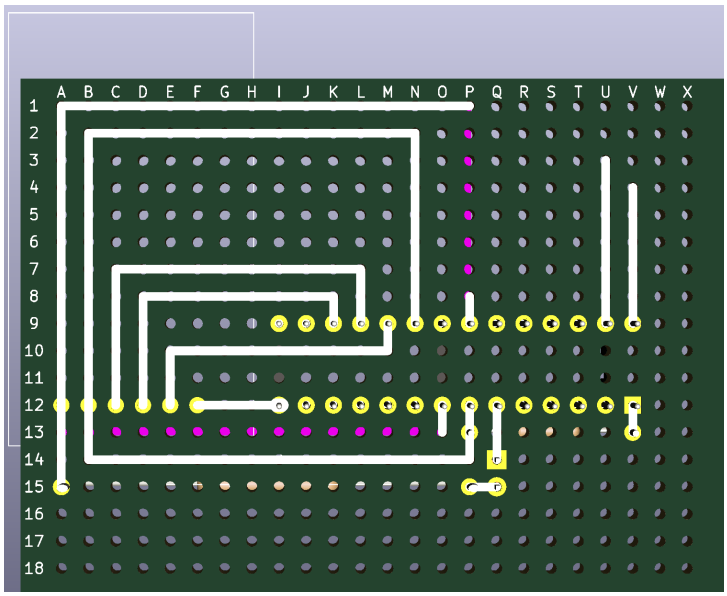


Board Assembly

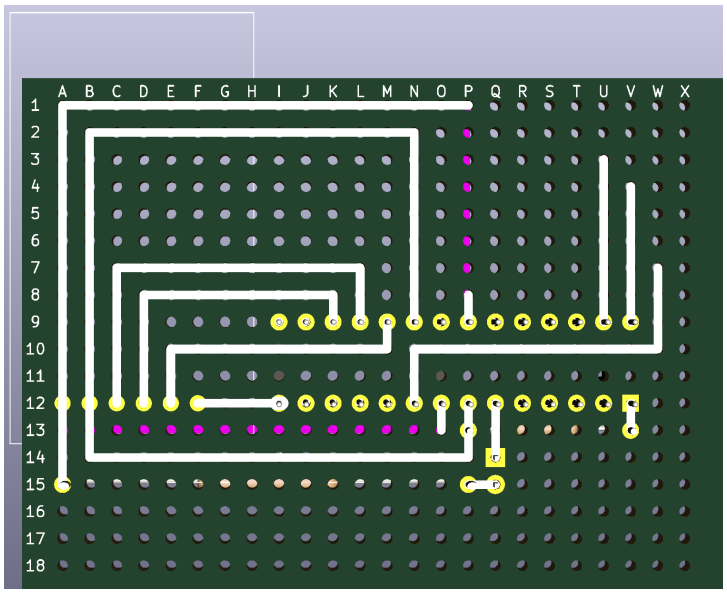
Step 22: Solder connections:
V9 to V4



Step 23: Solder connections:
U9 to U3

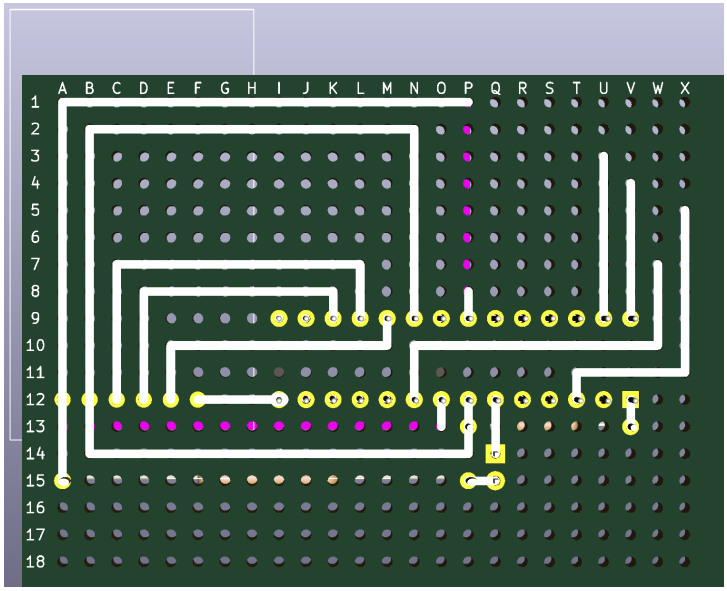


Step 24: Solder connections:
N12 to N10 to W10 to W7

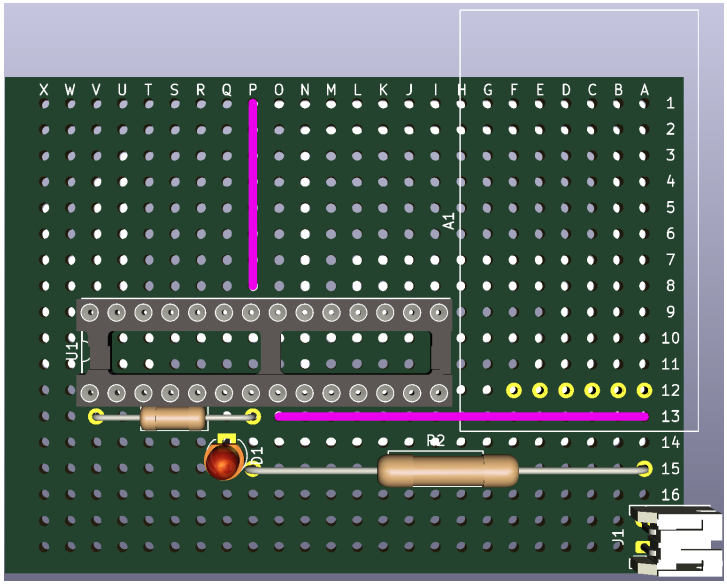


Board Assembly

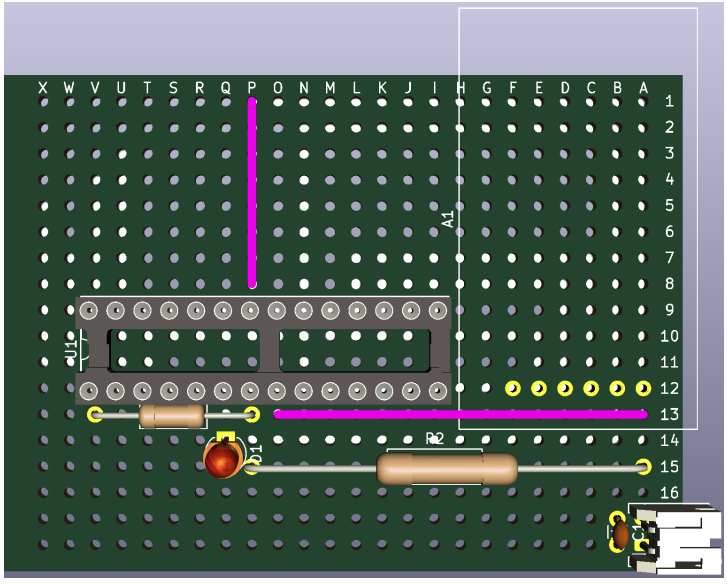
Step 25: Solder connections:
T12 to T11 to X11 to X5



Step 26: Place the 2 Pin Male JST Connector on pins A17 and A18 such that it is facing away from the board. The orientation of this piece DOES matter.

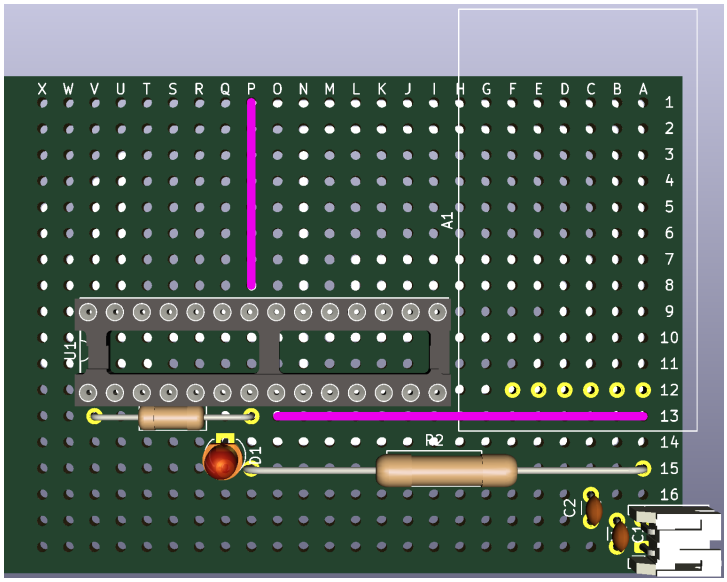


Step 27: Place a 1 µF Capacitor across pins B17 and B18. The orientation of this piece does not matter.

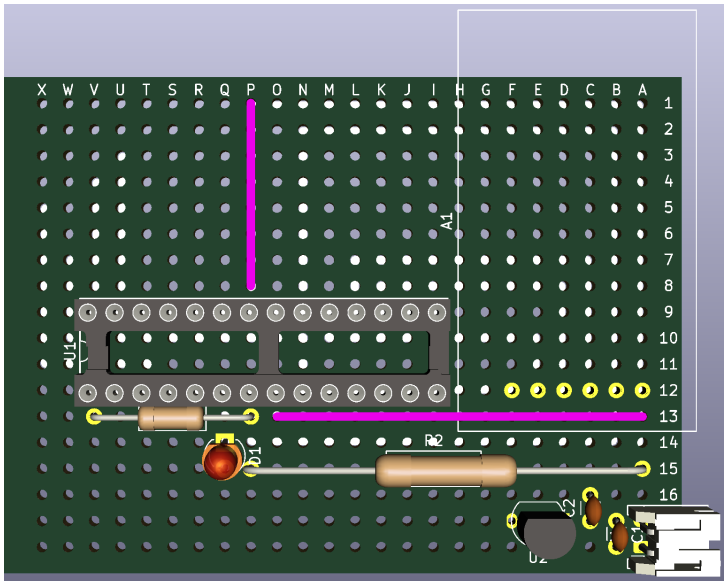


Board Assembly

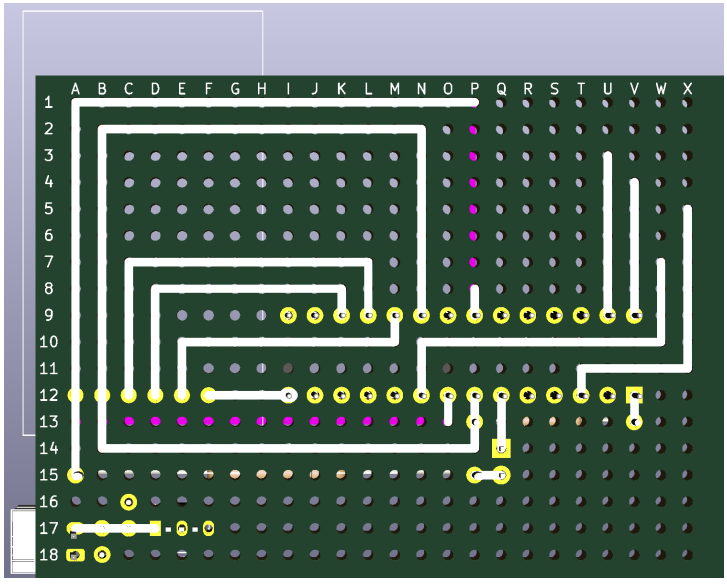
Step 28: Place the other 1 µF Capacitor across pins C16 and C17. The orientation of this piece does not matter.



Step 29: Place the 3V Linear Voltage Regulator on pins D17 (GND), E17 (V_{IN}), and F17 (V_{OUT}). The orientation of this piece DOES matter. The flat side of the component should face into the board.

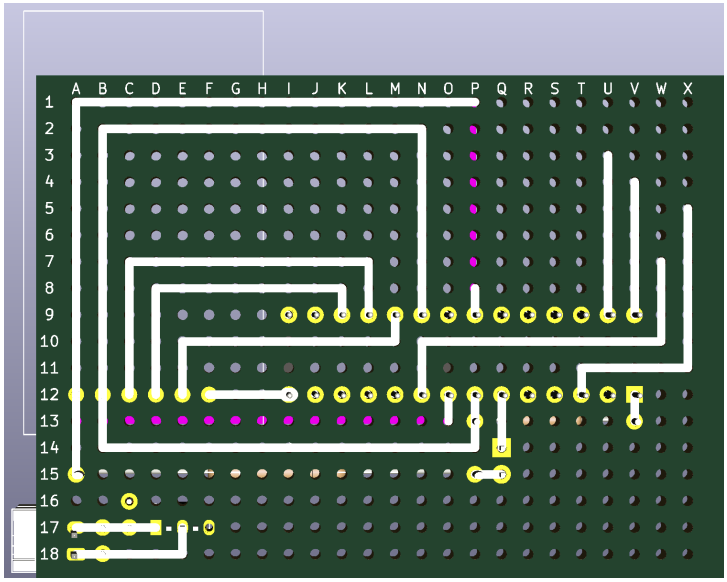


Step 30: Solder connections:
A17 to D17

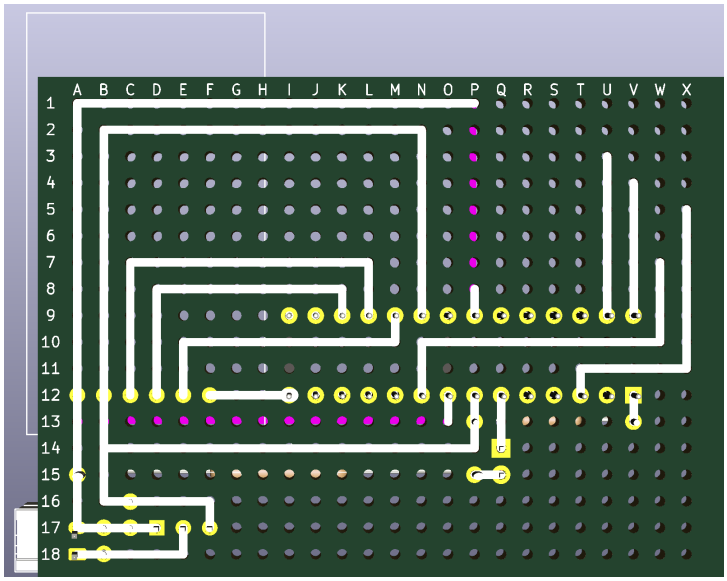


Board Assembly

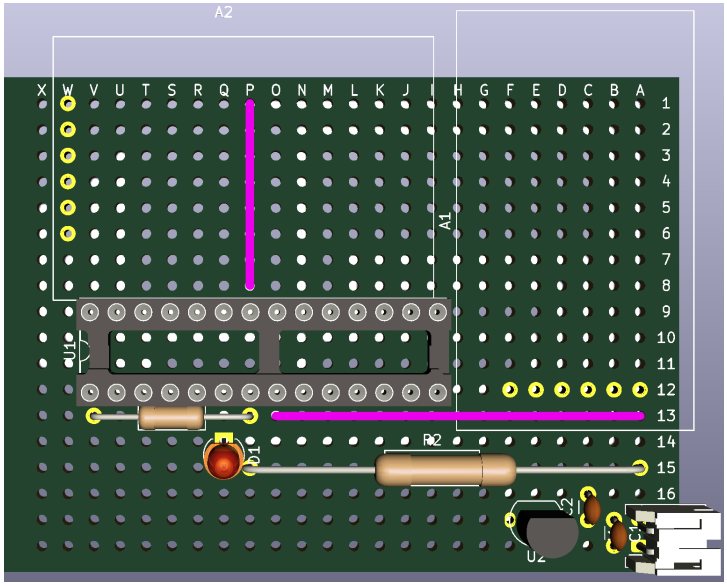
Step 31: Solder connections:
A18 to E18 to E17



Step 32: Solder connections:
F17 to F16 to B16 to B14

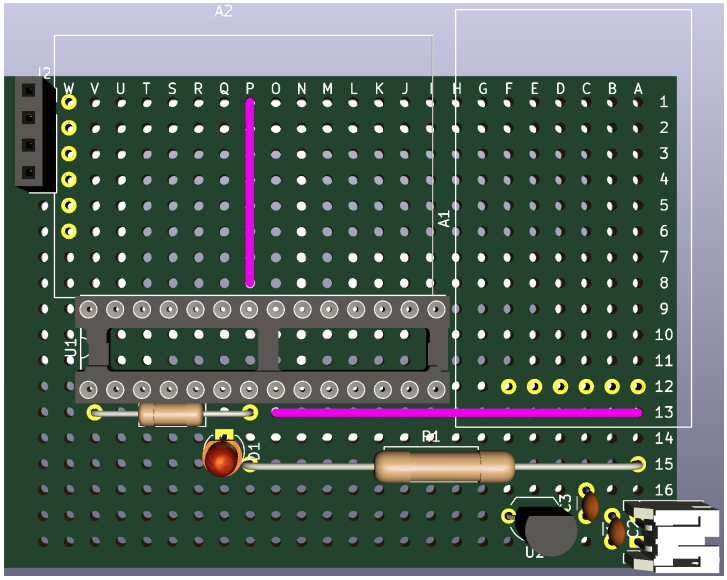


Step 33: Place the DS3231SN Breakout onto the perfboard:
GND goes through W1
VCC goes through W2
SDA goes through W3
SCL goes through W4
SQW goes through W5
32K goes through W6

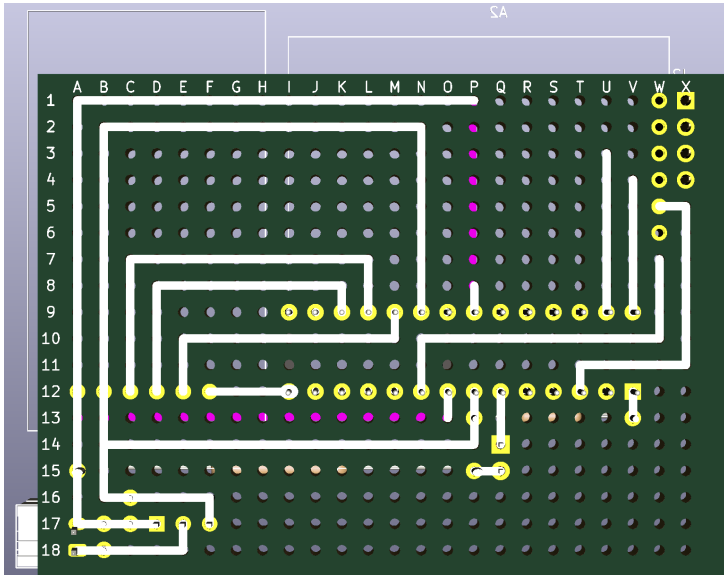


Board Assembly

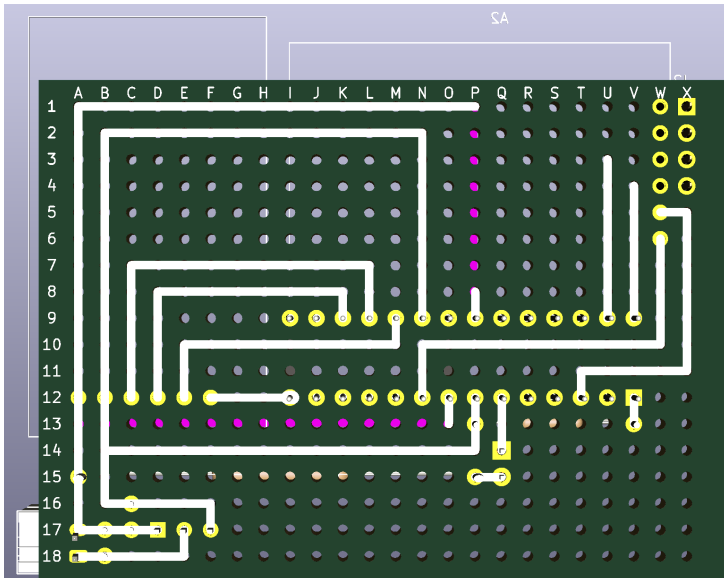
Step 34: Place a set of 4 Female Header Pins through X1, X2, X3, and X4. The orientation of this piece does not matter.



Step 35: Solder connections:
X5 to W5

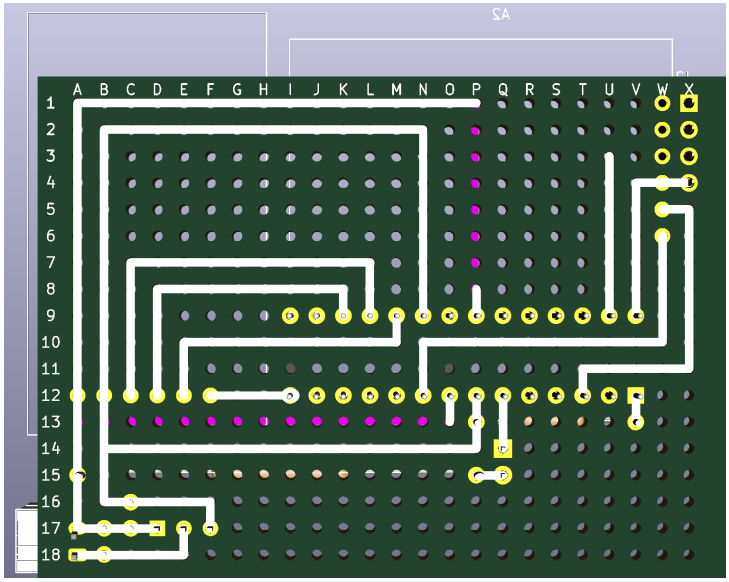


Step 36: Solder connections:
W7 to W6

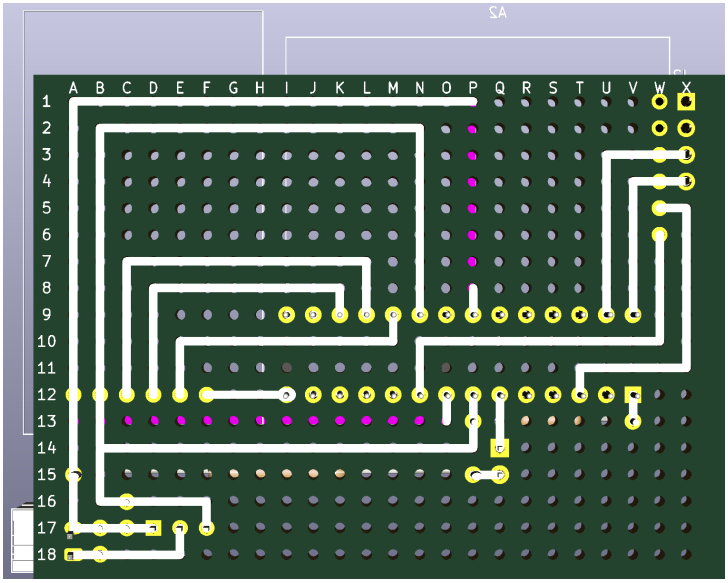


Board Assembly

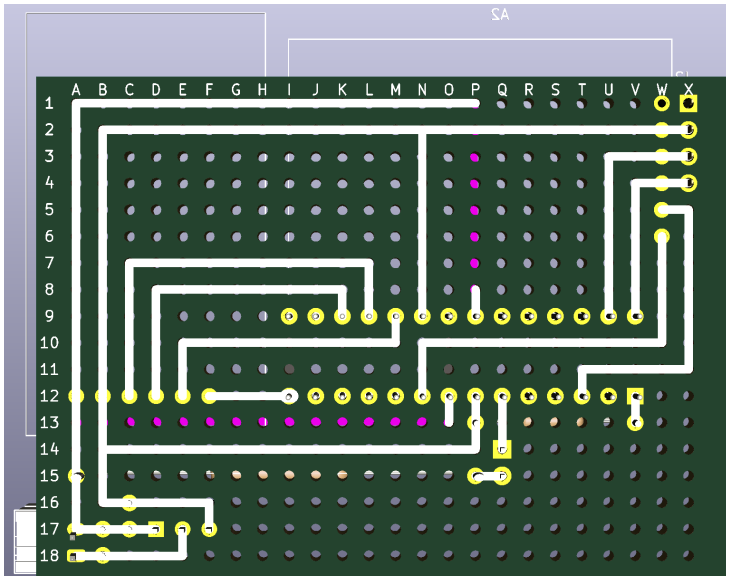
Step 37: Solder connections:
V4 to X4



Step 38: Solder connections:
U3 to X3

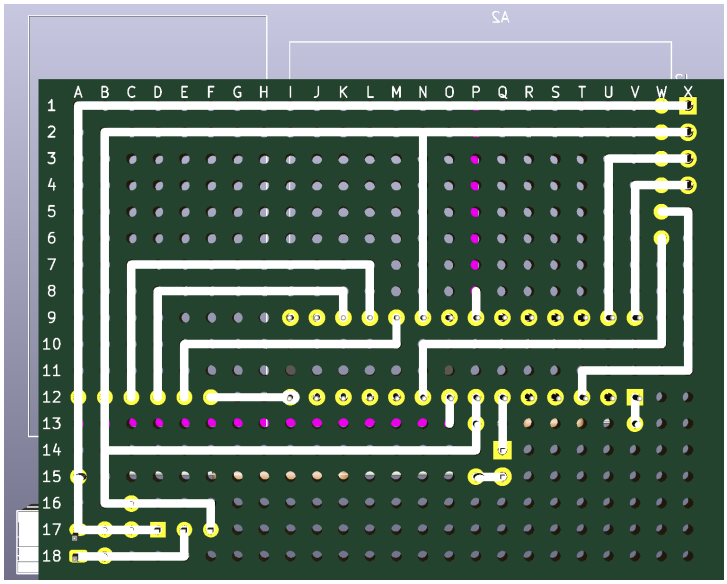


Step 39: Solder connections:
N2 to X2

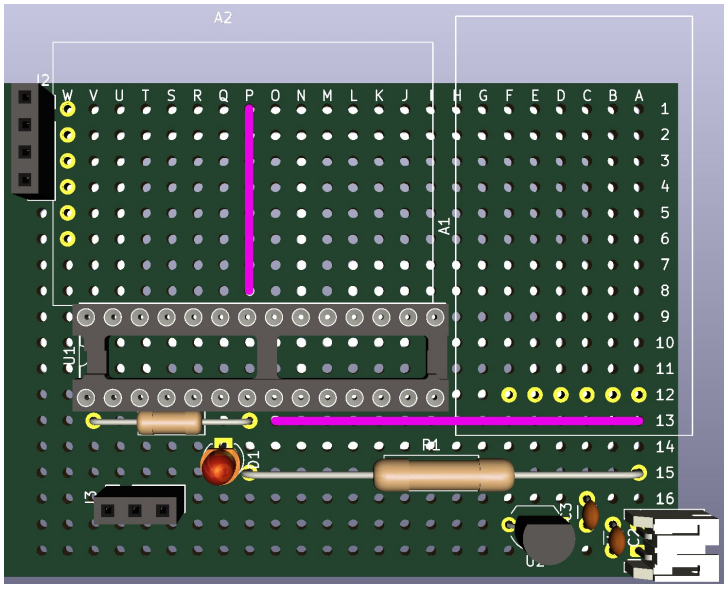


Board Assembly

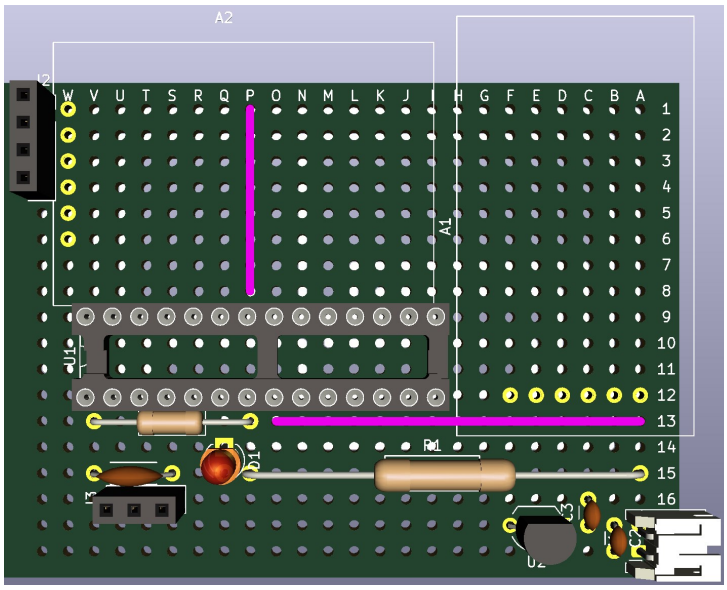
Step 40: Solder connections:
P1 to X1



Step 41: Place a set of 3 Female Header Pins through U16, T16, and S16. The orientation of this piece does not matter.

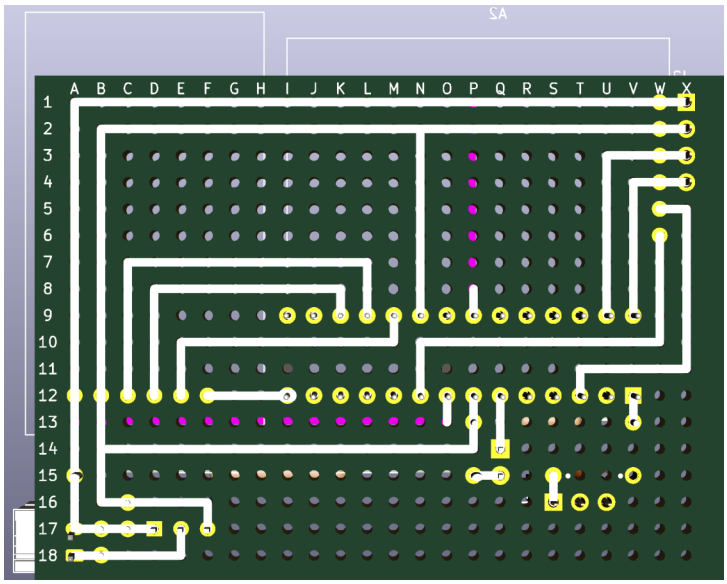


Step 42: Place a 0.1 µF Capacitor across pins S15 and V15. The orientation of this piece does not matter.

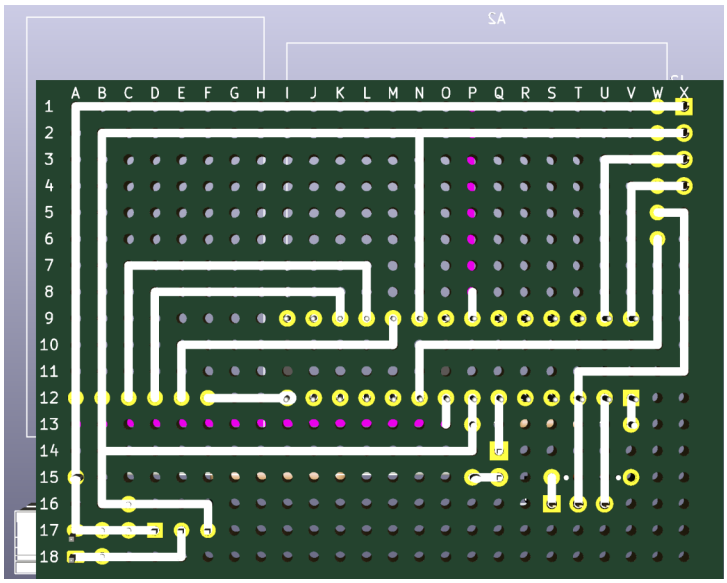


Board Assembly

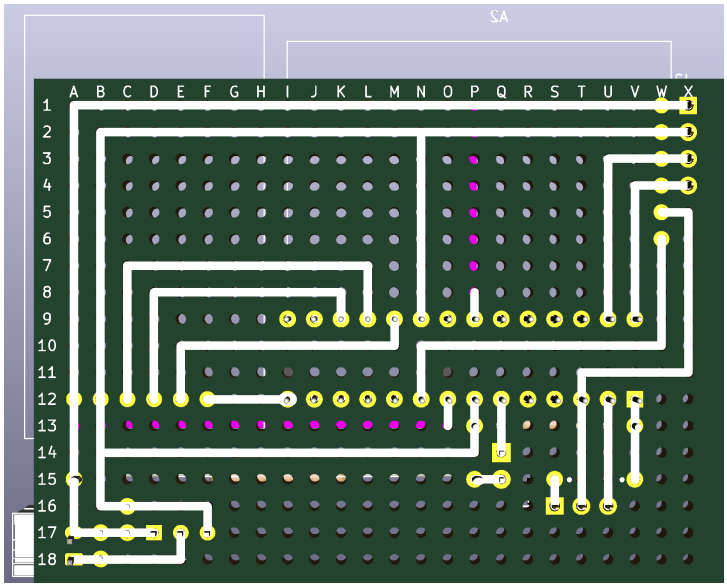
Step 43: Solder connections:
S16 to S15



Step 44: Solder connections:
T16 to T12
U16 to U12

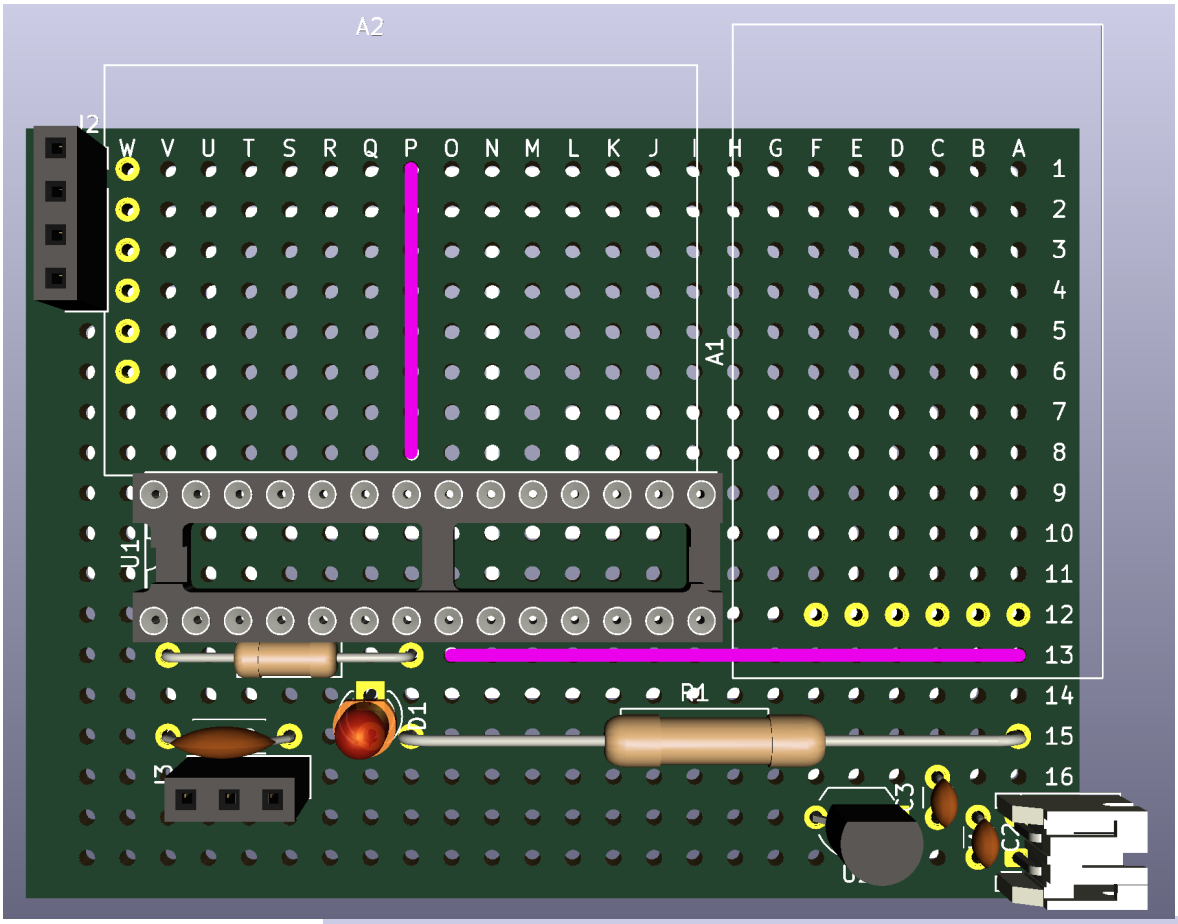


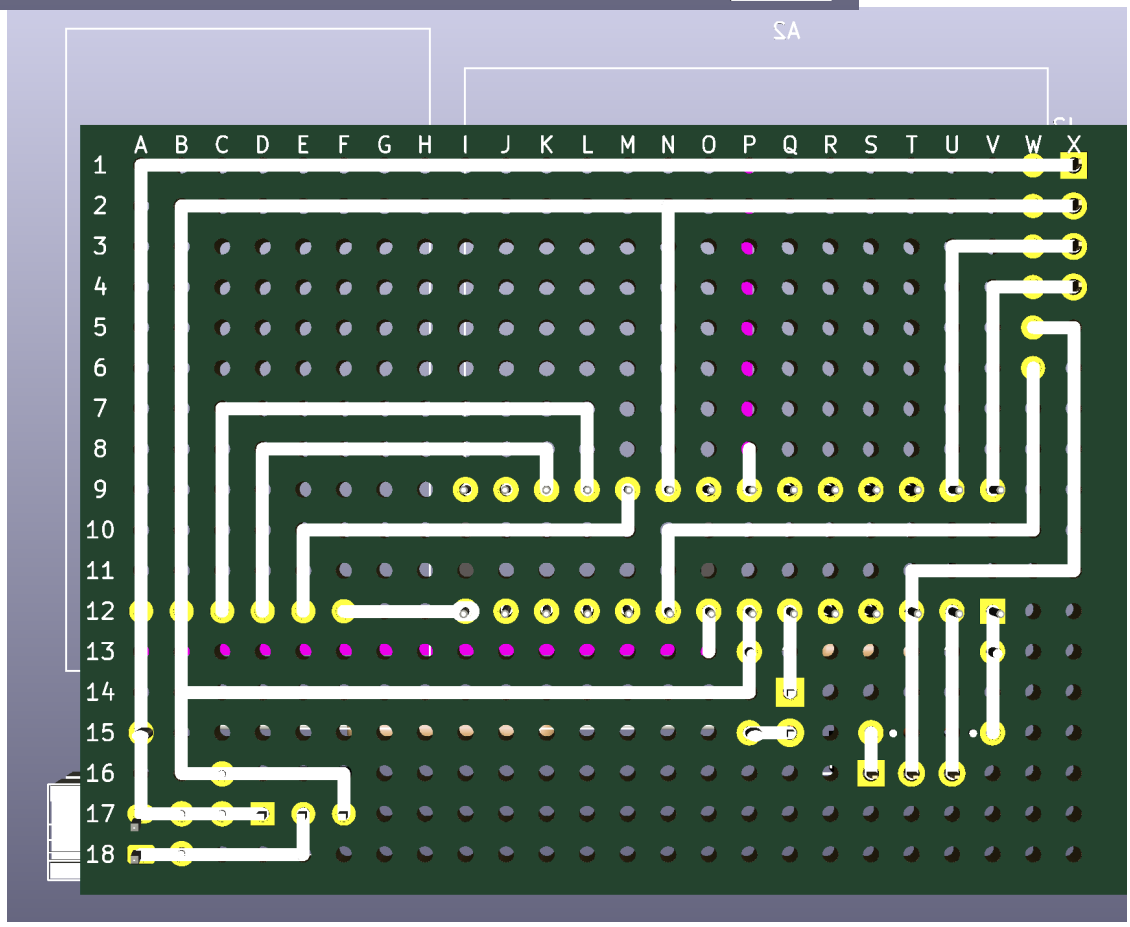
Step 45: Solder connections:
V15 to V13



Board Assembly

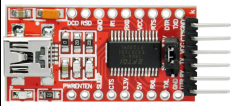
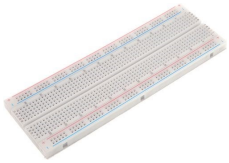
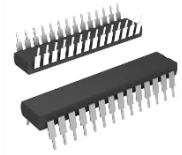
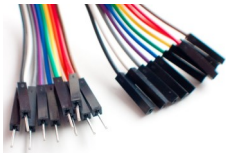
Assembly Complete!





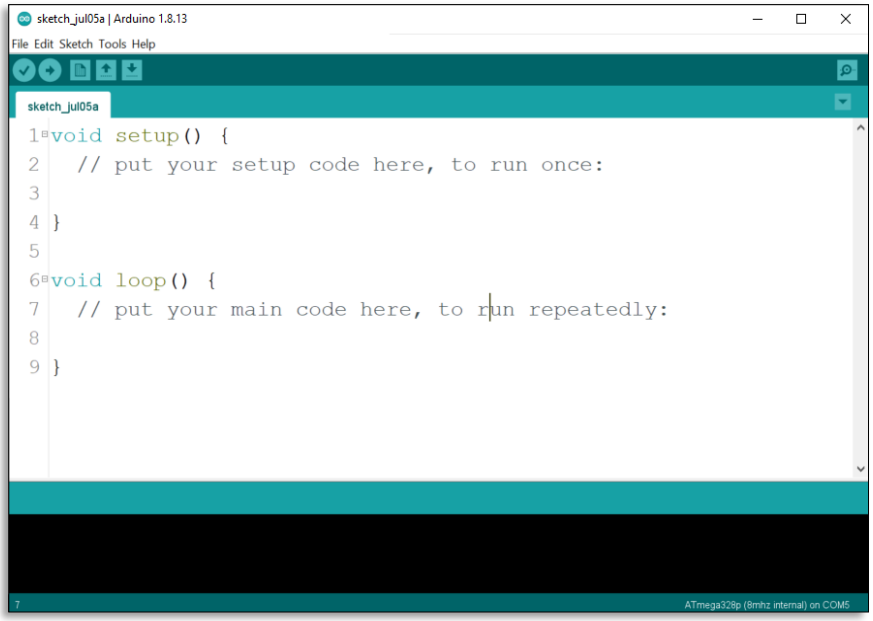
Programming the ATmega328P

This portion of the guide will explain how to upload a bootloader and code onto the ATmega328P chip using the Arduino IDE. These steps will require the following equipment:

Image	Part	Quantity	Image	Part	Quantity
	Arduino Uno	1		USB Type A to USB Type B Cable	1
	3.3V FTDI USB to TTL Serial Adapter	1		USB Type A to USB Type B Mini	1
	Breadboard	1		ATmega328P	1
	10 KΩ Resistor	1		1 µF Capacitor	1
	Male to Male, Male to Female, and Female to Female Jumper Wires	*depends on wiring*			

Downloading and Setting Up the Arduino IDE

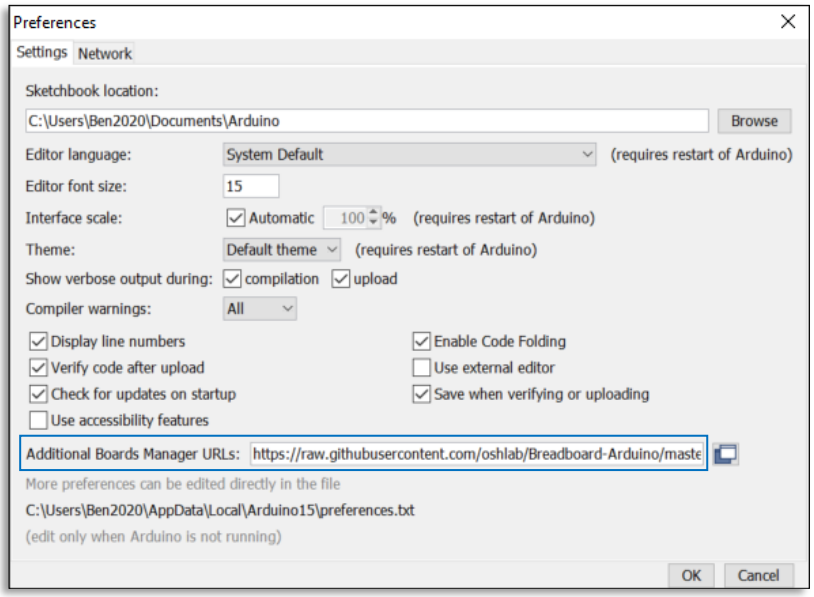
The Arduino IDE is the software used to develop code for and program Arduinos. Although this pressure logger does not use a full Arduino, it does use the same chip as the Arduino Uno (the ATmega328P). This means the user friendly and free Arduino IDE can still be used to interact with the standalone ATmega328P.



Programming the ATmega328P

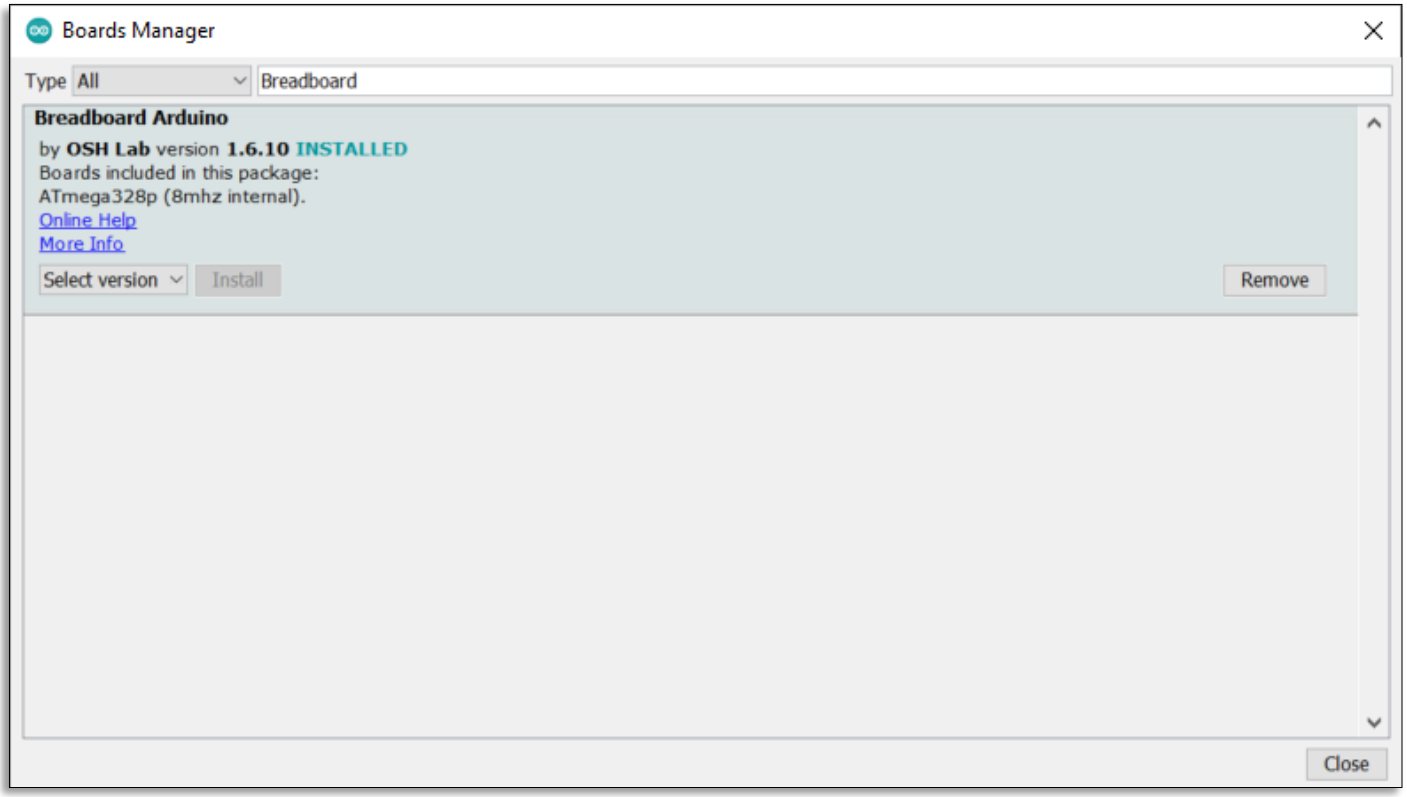
1. To download the IDE, navigate to https://www.arduino.cc/en/software and download the appropriate installer for your OS. See the "Install the Arduino Desktop IDE" section at https://www.arduino.cc/en/Guide for more details. This guide is based off of version 1.8.13 of the IDE. Once the IDE has been downloaded, install and run it. You should be greeted with a window similar to the one on the previous page.

2. In the IDE, go to `File -> Preferences` and paste the following link in the "Additional Boards Manager URLs" box:
https://raw.githubusercontent.com/oshlab/Breadboard-Arduino/master/avr/boardsmanager/package_oshlab_breadboard_index.json

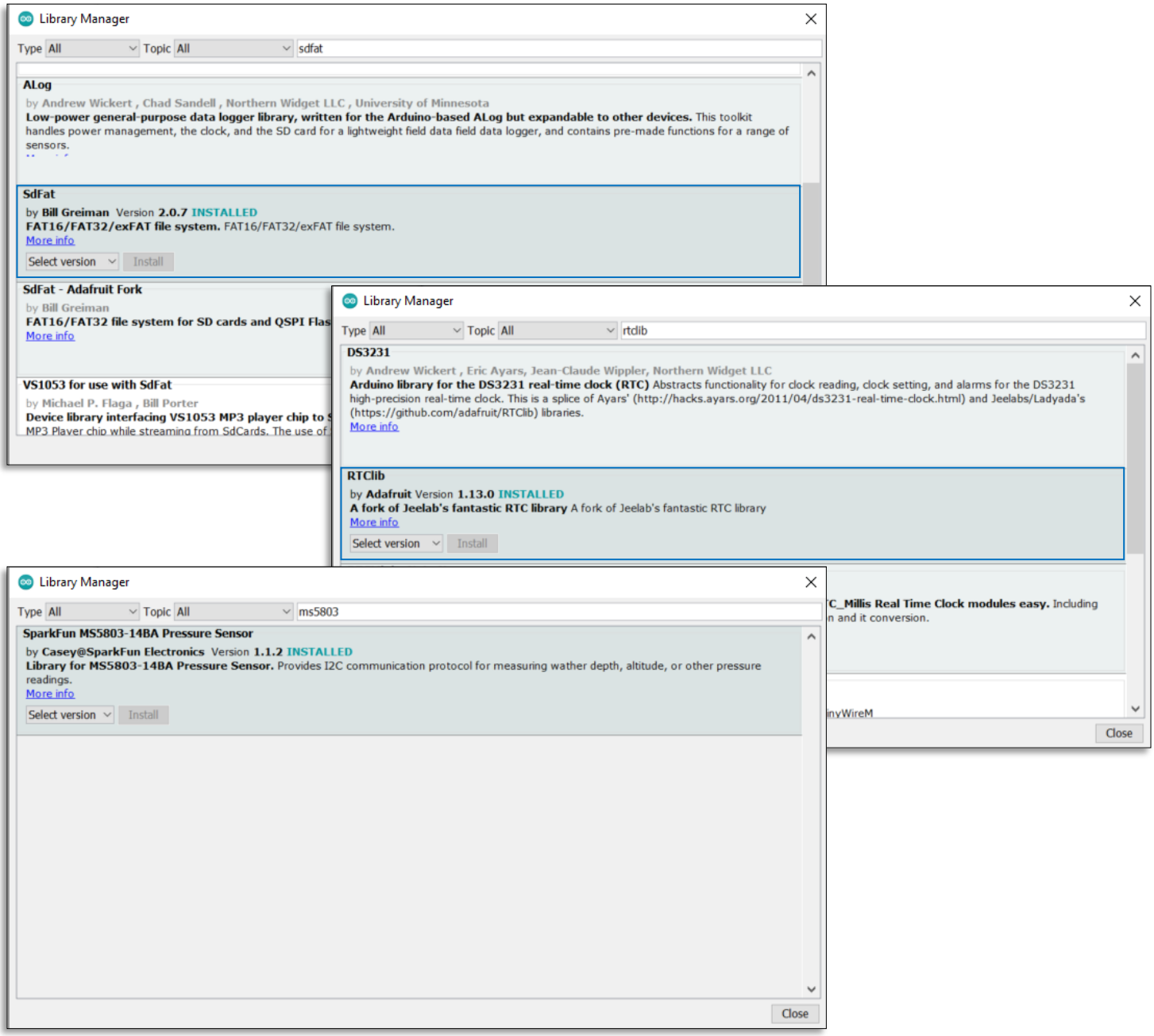


This link points to a board configuration that will be downloaded next for the ATmega328P, specifically one that uses the 8MHz internal clock of the chip. It can be found on the unofficial list of 3rd party Arduino boards at https://github.com/arduino/Arduino/wiki/Unofficial-list-of-3rd-party-boards-support-urls. This allows us to run the chip at 3V. The other settings here can be customized.

3. Navigate to `Tools -> Board -> Boards Manager`. Find the "Breadboard Arduino" configuration (search for "Breadboard") and install it.



4. Navigate to `Sketch -> Include Library -> Manage Libraries`. Three external libraries are needed: "SdFat", "RTClib", and "SparkFun MS5803-14BA Pressure Sensor". Find each one in the library manager and install it. The version of each library used by this guide is shown in the images below; if the most recent version of the library causes issues during later steps, try using the versions shown here.

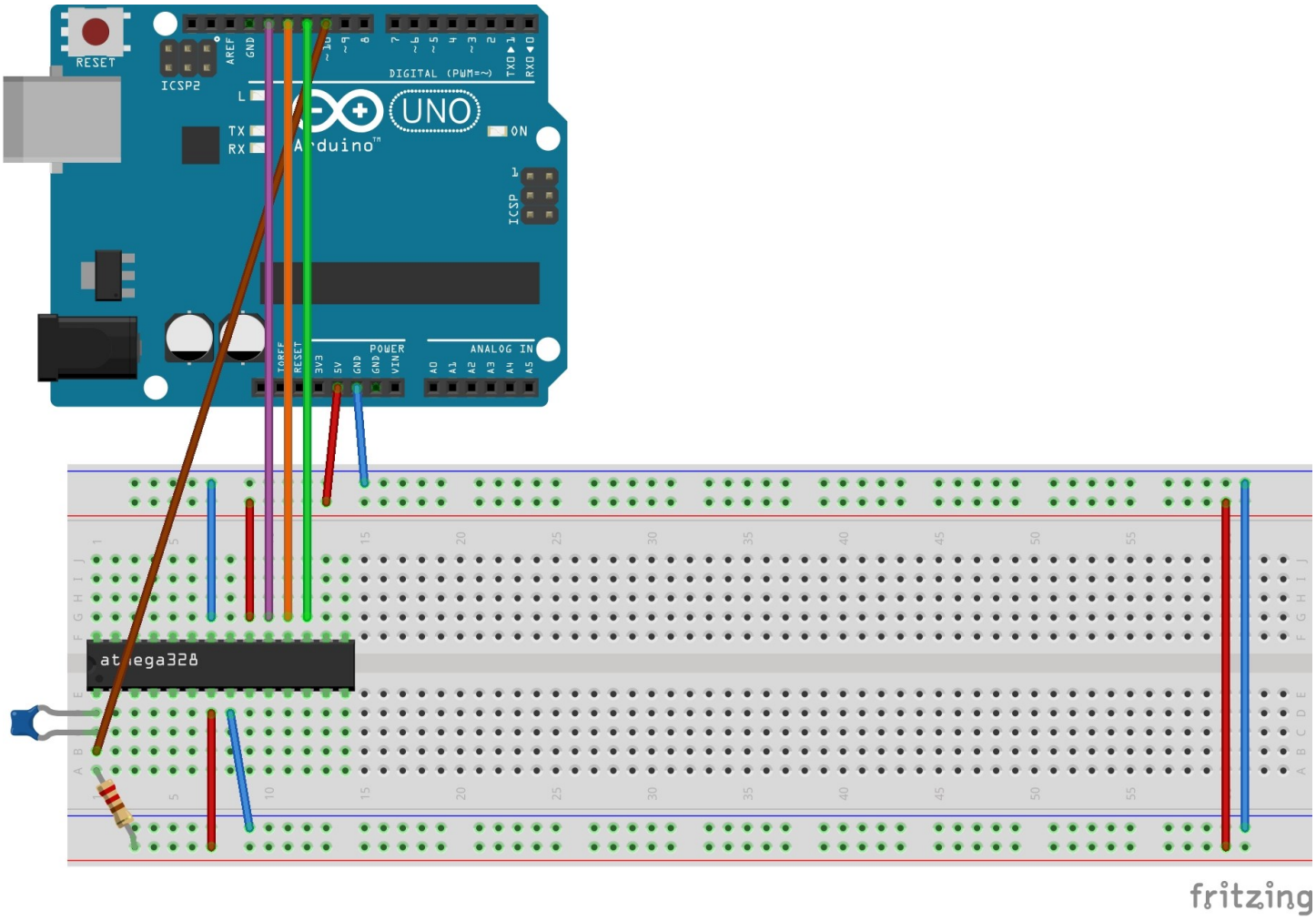


Programming the ATmega328P

Uploading the Bootloader

- This step is only required for fresh ATmega328P chips. After the bootloader has been burned into the chip once, it will not need to be burned again.
- There are many ways to burn a bootloader onto a chip, and this guide only briefly describes one possible approach.

First, the ATmega328P must be set up on a breadboard. This breadboard setup will allow the "ATmega328p (8mhz internal)" bootloader to be burned onto the ATmega328P using an Arduino. The following image shows the necessary wiring. The blue capacitor has a value of 1 µF and the resistor has a value of 10 KΩ; their orientations do not matter. Do note the orientation of the ATmega328P, however, as the notched end must face away from the breadboard (as shown dimly in the image).

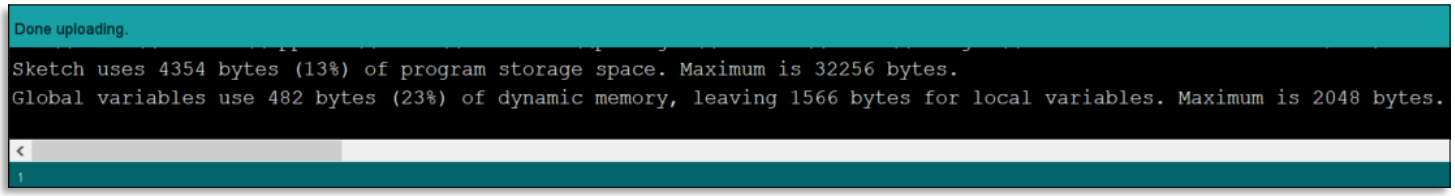


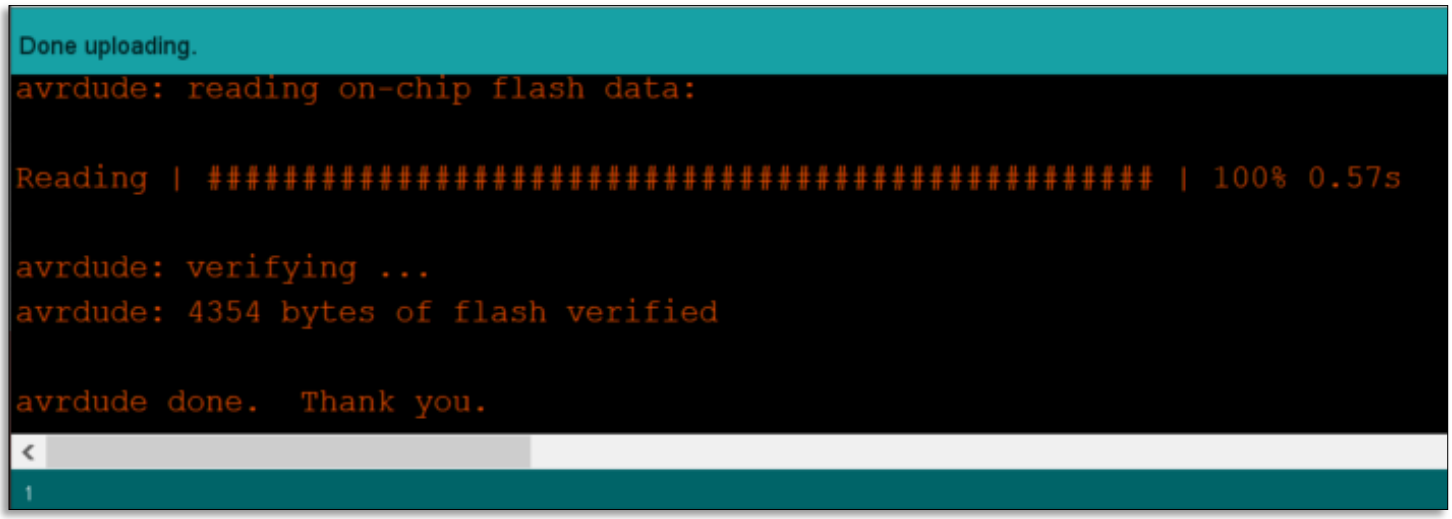
Double check all wiring before plugging the Arduino Uno into your computer using the USB Type A to USB Type B Cable.

Programming the ATmega328P

The Arduino IDE will be used for the following steps:

1. In order to use the Arduino Uno to upload a bootloader, the Uno must be programmed with the "ArduinoISP" sketch. Navigate to `File -> Examples -> ArduinoISP` and select `ArduinoISP` to open the desired sketch. Use the new window for the remaining steps.
2. Navigate to `Tools -> Board -> Arduino AVR Boards` and select `Arduino Uno`.
3. Navigate to `Tools -> Port` and select the port with the Arduino Uno attached.
4. Navigate to `Tools -> Programmer` and select `AVRISP mkII`.
5. Upload the sketch using the "Upload" button found at the top of the window or in the `Sketch` menu. The output should look similar to either of the following:



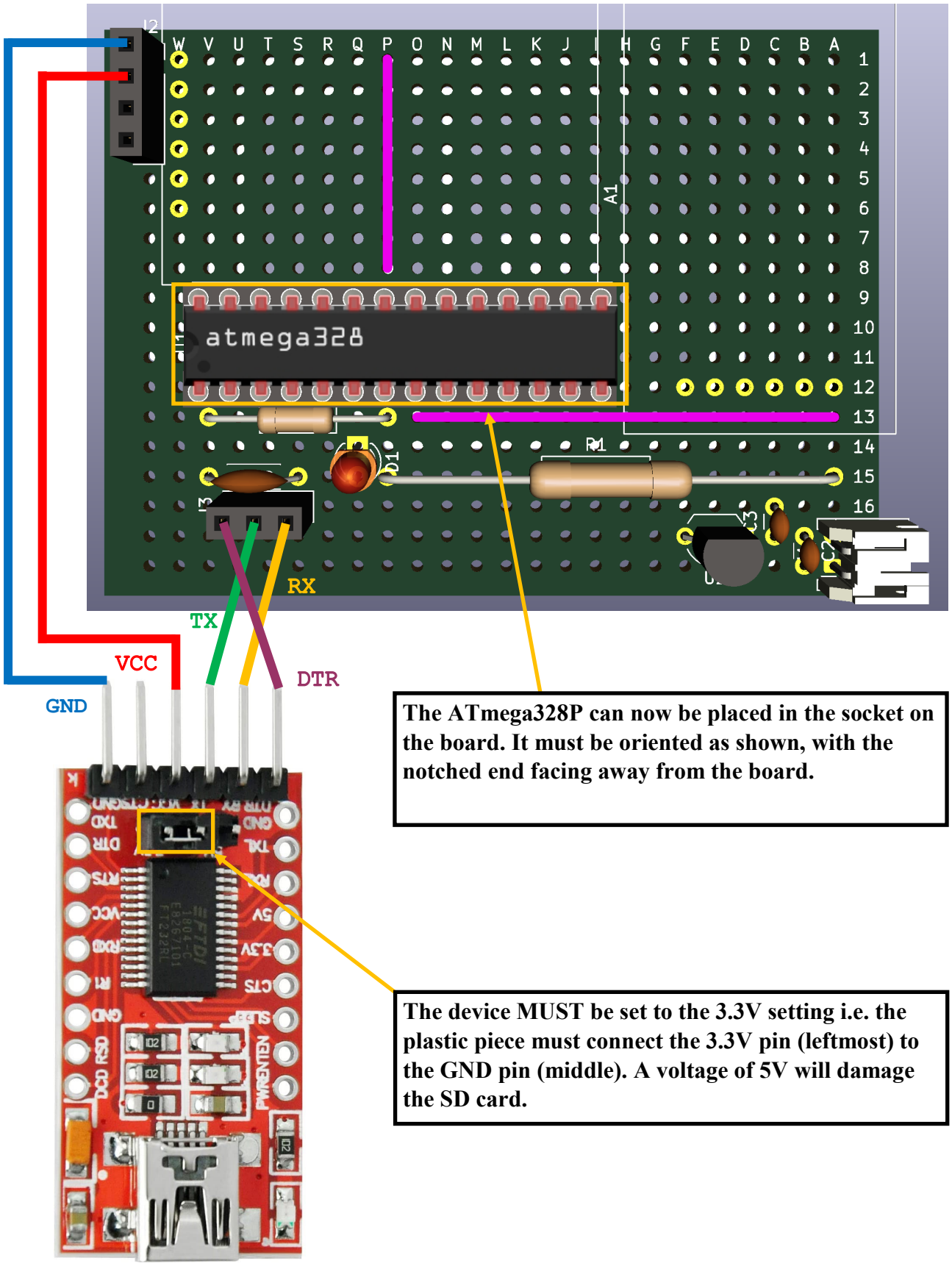


6. Now that the Uno is ready, the bootloader can be uploaded. Navigate to `Tools -> Board -> Breadboard Arduino (OSHLAB.com)` and select `ATmega328p (8mhz internal)`.
7. Navigate to `Tools -> Programmer` and select `Arduino as ISP`.
8. Burn the bootloader to the chip by pressing the "Burn Bootloader" button in the `Tools` menu. If the operation is successful, a light will blink once per second on the Uno.

Programming the ATmega328P

Uploading Code

The following circuit must be set up in order to upload code to the ATmega328P. Once ready, one program will be uploaded to set the time on the RTC before uploading the main code used by the device.



The ATmega328P can now be placed in the socket on the board. It must be oriented as shown, with the notched end facing away from the board.

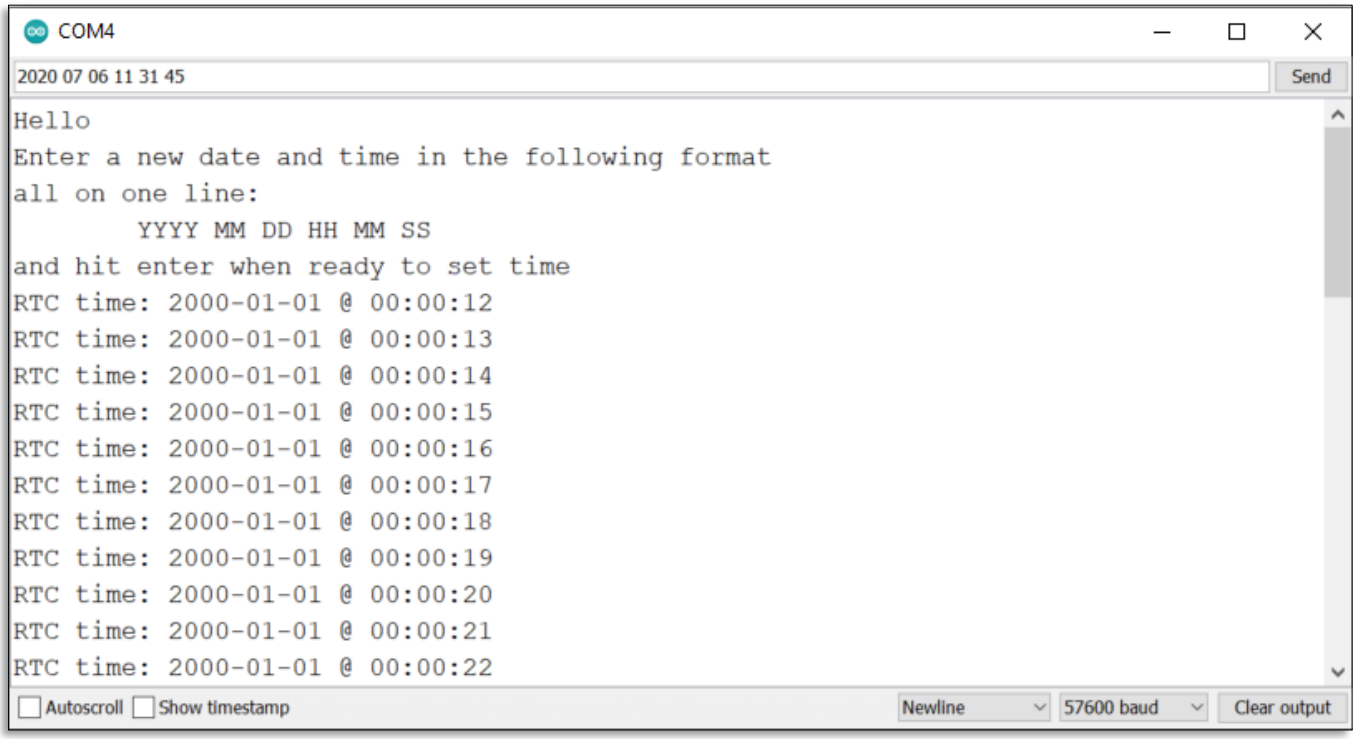
The device MUST be set to the 3.3V setting i.e. the plastic piece must connect the 3.3V pin (leftmost) to the GND pin (middle). A voltage of 5V will damage the SD card.

Programming the ATmega328P

Two sketches are needed, one for setting the time and another for the main program.

Setting the Time:

1. Make sure the RTC module on your board is fitted with a CR2032 coin cell battery. This allows the module to continue keeping the time even after power is removed.
2. An updated version of Luke Miller's `settime_Serial` sketch found in his OWHL repository can be used to set the time, and can be found here. Download it (CTRL+S) as "settime_Serial.ino" and save it into a new folder named "settime_Serial" (required for the Arduino IDE to function properly).
3. Open the sketch in the Arduino IDE and connect the 3.3V FTDI USB to TTL Serial Adapter to your computer using the USB Type A to USB Type B Mini cable.
4. Navigate to `Tools -> Board -> Breadboard Arduino (OSHLAB.com)` and select `ATmega328p (8mhz internal)` if it is not already selected.
5. Navigate to `Tools -> Port` and select the port with the 3.3V FTDI USB to TTL Serial Adapter attached. This port may not be labelled (it may simply be called COM4, for example), and you may have to try uploading the code to each port until the correct one is found.
6. Navigate to `Tools -> Programmer` and select `AVRISP mkII`.
7. Upload the sketch using the "Upload" button found at the top of the window or in the `Sketch` menu. If errors occur, check all wiring and configuration settings from above.
8. Open the Serial Monitor in the Arduino IDE by pressing the magnifying glass icon or navigating to `Tools -> Serial Monitor`. It should begin outputting the current time according to the RTC. This time will most likely be incorrect.
9. Set the time by typing into the field at the top of the Serial Monitor. Enter the current date and time in the following format: `YYYY MM DD hh mm ss` and click send or press Enter. The time being outputted in the monitor should be updated. This step can be repeated for accuracy.



Programming the ATmega328P

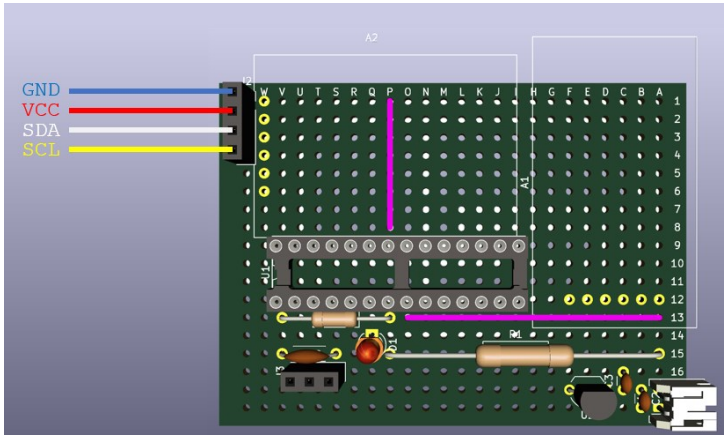
Uploading the main sketch:

1. The main sketch can be found here. Download it (CTRL+S) and save it as DIY2.ino in a new folder called DIY2.
2. Open the sketch in the Arduino IDE and make sure the 3.3V FTDI USB to TTL Serial Adapter is still connected to your computer and the circuit board.
3. Navigate to `Tools -> Board -> Breadboard Arduino (OSHLAB.com)` and select `ATmega328p (8mhz internal)` if it is not already selected.
4. Navigate to `Tools -> Port` and select the port with the 3.3V FTDI USB to TTL Serial Adapter attached. This port may not be labelled (it may simply be called COM4, for example), and you may have to try uploading the code to each port until the correct one is found.
5. Navigate to `Tools -> Programmer` and select `AVRISP mkII`.
6. Upload the sketch using the "Upload" button found at the top of the window or in the `Sketch` menu. If errors occur, check all wiring and configuration settings from above. Otherwise, the red LED on the circuit board should light up, indicating possible errors which will be addressed in the next section, or that the sensor is now sampling.

Operation Details

Initialization

Once the housing and the circuit board have been assembled, the ATmega328p microchip has been programmed, and the DS3231SN Real Time Clock (RTC) has been set to the correct time, the pressure sensor is ready to collect data. Before powering a sensor, ensure the DS3231SN Breakout is fitted with a CR2032 coin cell battery, the microSD Breakout has a microSD card in the slot, and the wires from the MS5803-14BA pressure sensor are plugged into the correct ports on the board, as shown on the right.



When the device is powered by plugging a battery into the connector in the lower right, it will immediately initialize the RTC, microSD card, and MS5803-14BA pressure sensor; there is no on/off switch. If any of these initializations fail, the device will stop and the error LED will blink at a rate indicating the issue:

- 1 blink repeating: issue with RTC
- 2 blinks repeating: issue with microSD card / breakout
- 3 blinks repeating: failed to create file on SD card

Warning blinks may also be seen:

- 1 short blink indicates that the RTC has lost power at some point, and needs to be reset to the correct time. The device will still function properly, but its time will not be accurate and will not be synchronized with other sensors.
- 2 short blinks indicate that there was an issue reading data from the configuration file on the SD card (see below), and that the device will be using default (hardcoded) settings.

Creating a Configuration File

The device has 4 configurable settings which can be modified by creating a `config.txt` file in the root directory of the SD card. This makes it easier to modify deployment parameters, as it does not require the sensor to be reprogrammed in order to make the changes. The settings are the following:

`startMinute`: The minute-hand value when the sensor should begin taking the first sample. A value greater than 59 will cause the sensor to begin sampling immediately. This can be used to start multiple sensors at the same time, assuming their clocks are accurately synchronized. The device goes into a low power sleep while waiting.

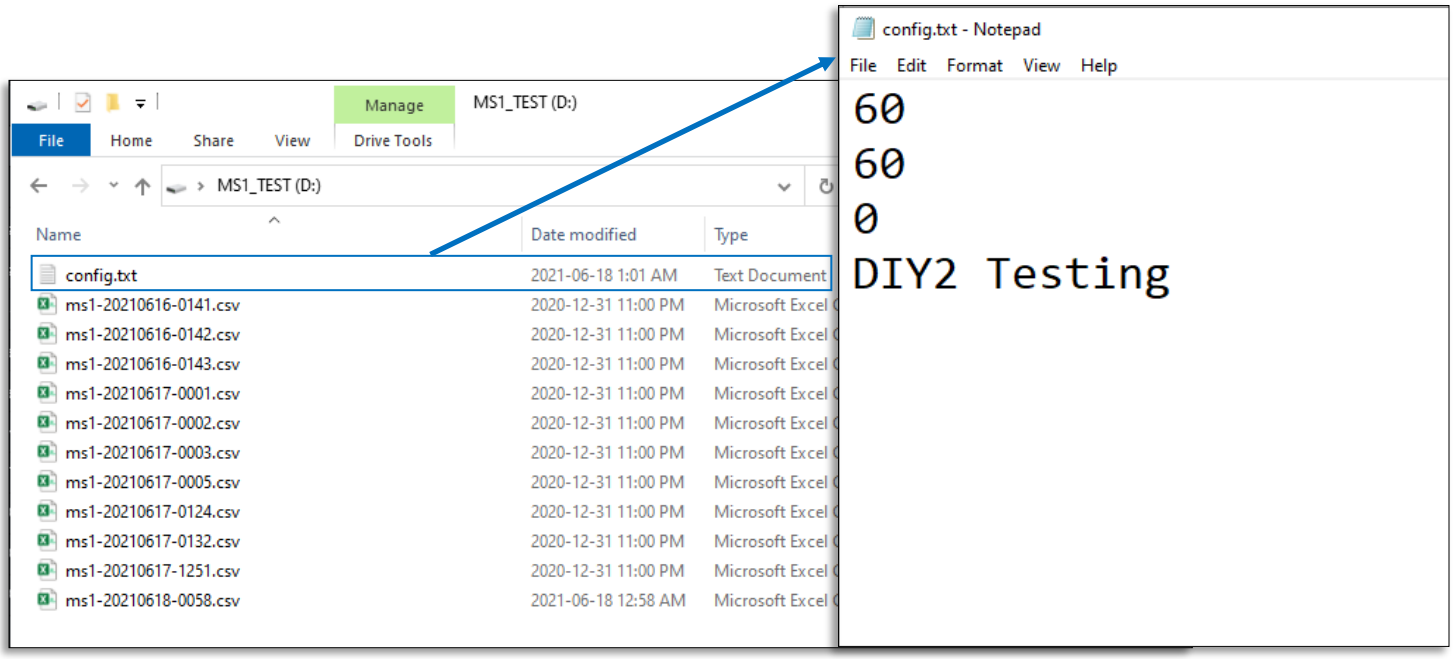
`sampleDuration`: The duration of time (in minutes) spent sampling before the device should sleep during a single cycle. This value will have no effect if `sleepDuration` is set to 0, since the device will be sampling continuously.

`sleepDuration`: The duration of time (in minutes) to spend in a low power sleep before resuming sampling during a single cycle. If the value is set to 0, the device will sample continuously (no deep sleeping).

`infoString`: A short piece of text to place in each CSV file. This can be used to identify the specific sensor and site (63 characters max).

Operation Details

The below example of a configuration file shows the following: `startMinute=60`, `sampleDuration=60`, `sleepDuration=0`, `infoString="DIY2 Testing"`. This means the sensors will begin sampling immediately after being powered, and will sample continuously (without sleeping). The text "DIY2 Testing" (without quotes) will be present in the second row of each CSV file created by the device. Note that only the values are present in the text file, and each value is separated by a new line in the order described above (`startMinute`, `sampleDuration`, `sleepDuration`, `infoString`).



Sampling

After the device has been initialized and the start time has been reached (if a start time was set), the device is ready to begin sampling. The LED will then flash for 3 seconds to notify the user. At the beginning of sampling and at the start of each day, a new CSV file will be created on the SD card. Depending on the `sleepDuration` value, the device may sample continuously or temporarily go to sleep after the time specified by `samplingDuration`. For example, using `samplingDuration=17` and `sleepDuration=3`, the device will first sample for 17 minutes, then go into a low power sleep for 3 minutes, then repeat. The device will remain active until the voltage provided by the battery drops sufficiently low or the power is removed.