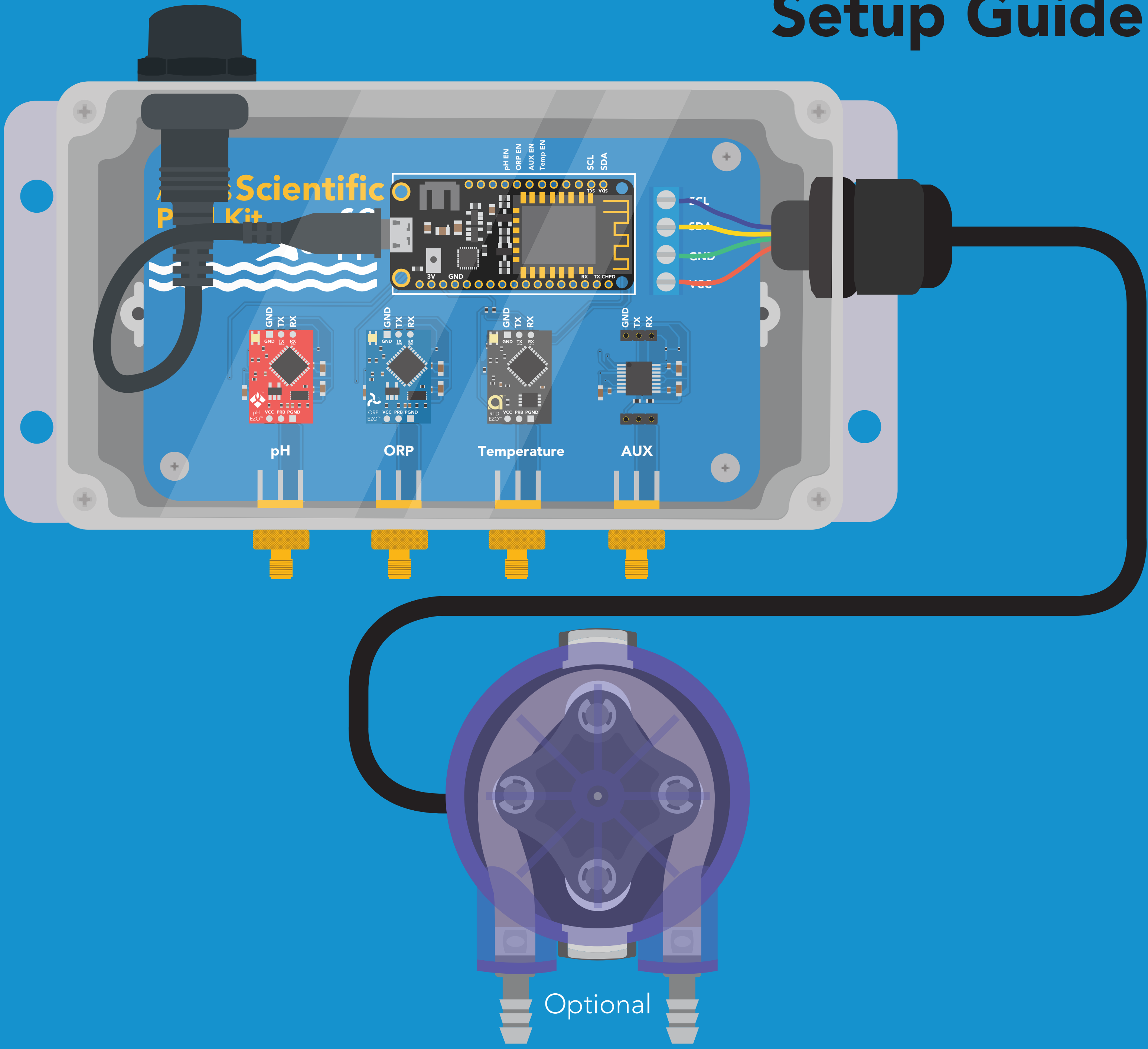
Atlas Scientific™
Environmental Robotics

V 1.3

Wi-Fi Pool Kit

Setup Guide



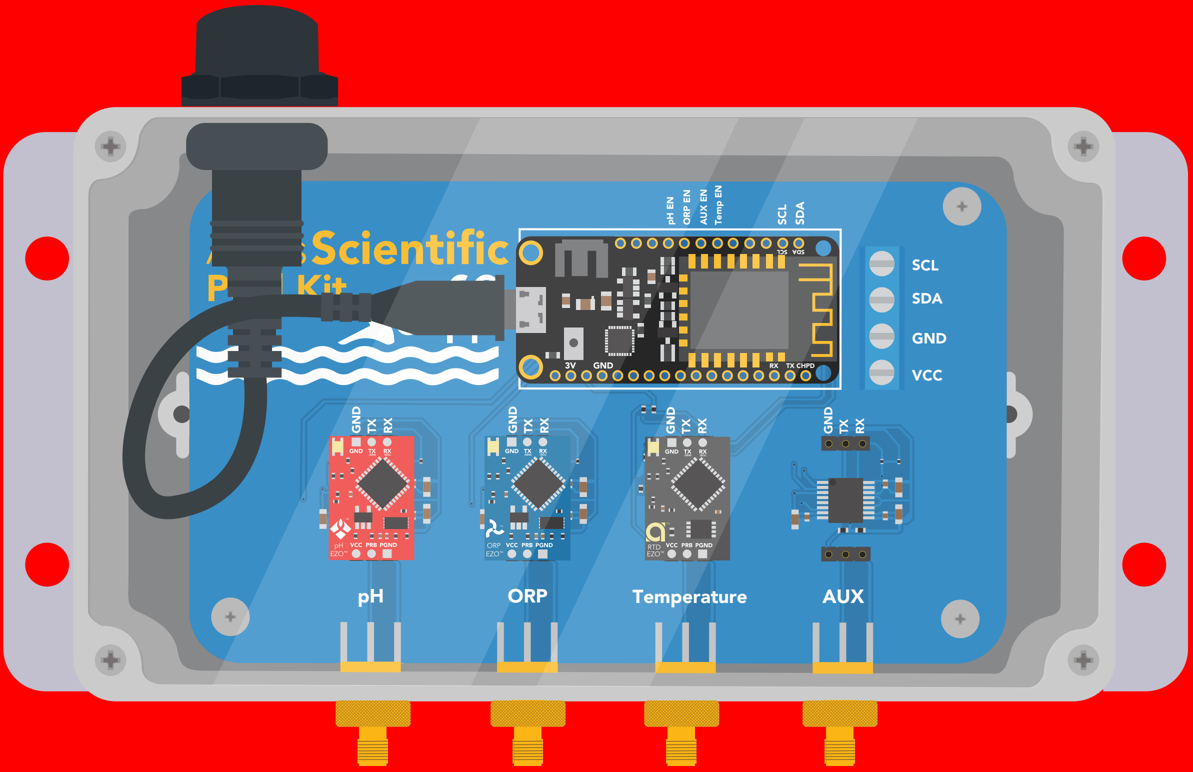
Optional

STOP

Atlas Scientific does not make consumer electronics.

This equipment is intended for electrical engineers. If you are not familiar with electrical engineering or embedded systems programing, this product may not be for you.

This device was developed and tested using a Windows computer. It was not tested on Mac, Atlas Scientific does not know if these instructions are compatible with a Mac system.



IP64

(dust and water splash proof)

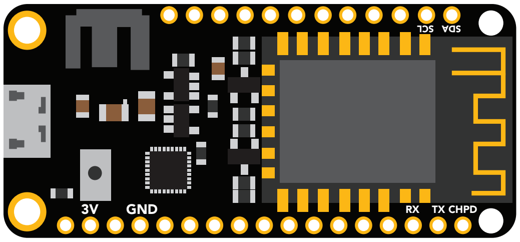
Operating principle

The Wi-Fi Pool Kit has been designed to provide the engineer with a simple way of remotely monitoring and controlling a pools system's chemistry. Sensor data is uploaded to ThingSpeak ™, a free, cloud-based data acquisition and visualization platform. The Wi-Fi Pool Kit has also been designed to be easily modified by the engineer. Feel free to change the sensors or functionality of the device to meet your specific needs.

Overview

CPU

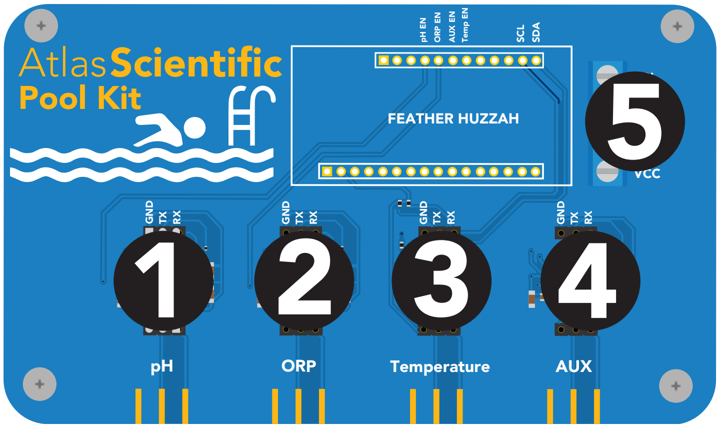
The Wi-Fi Pool Kit is controlled using an Adafruit Feather HUZZAH as its CPU. The HUZZAH is programmed using the Arduino IDE and uses an onboard ESP8266 as its Wi-Fi transmitter. [Adafruit Feather HUZZAH datasheet.]()



Sensor ports

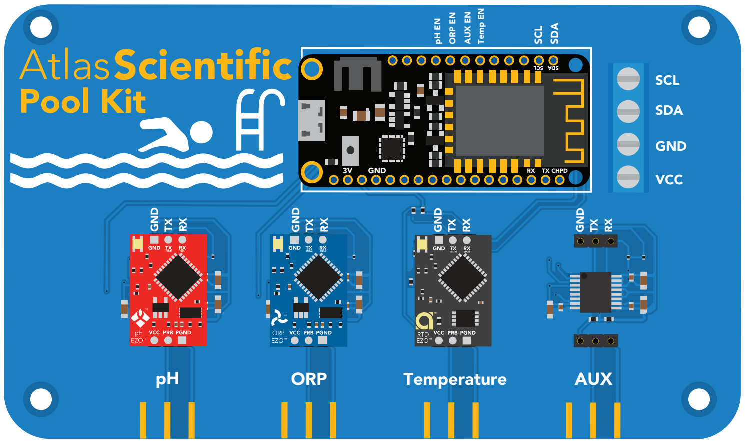
The Wi-Fi Pool Kit PCB has 5 sensor ports. Three of the ports are electrically isolated. The isolated ports are marked pH, ORP, and AUX. The isolated ports are needed to take noise-free electrochemical readings. Because the sensing element of a temperature sensor is never in direct contact with the water, electrical isolation is not needed for temperature sensing.

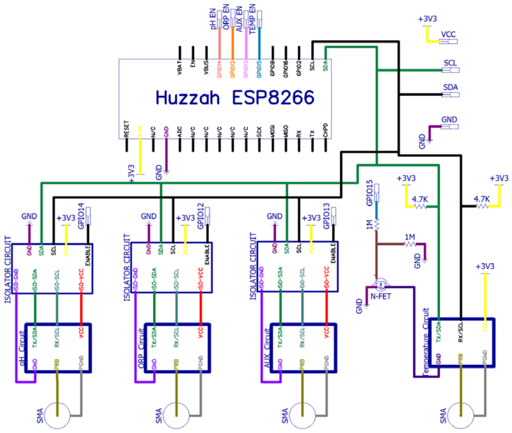
The AUX port can be used to add an additional sensor of your choice. The terminal block marked Port 5 has been designed to connect one or more dosing pumps to the device. However, the port could also be used to connect a gas sensor.



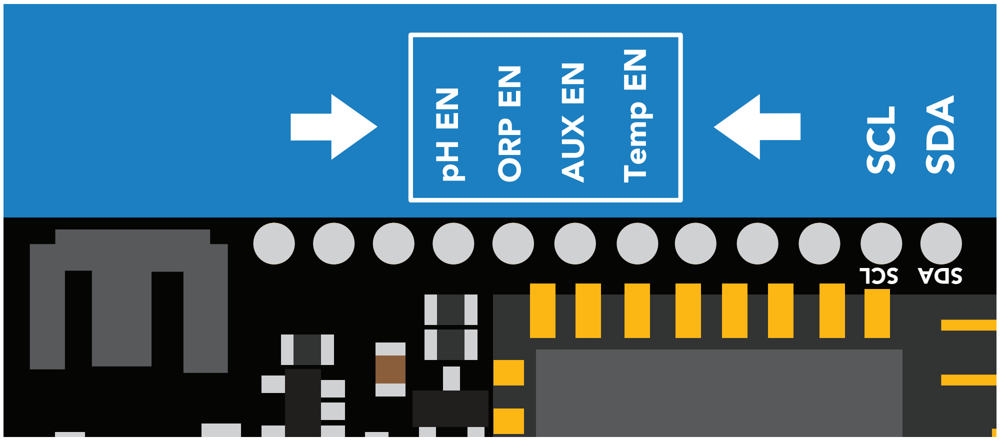
PCB

The overall design of the PCB is quite simple. The CPU is powered and programmed through the panel-mount USB connector. The CPUs onboard voltage regulator supplies the board's power bus with 3.3V at 500 mA peak. All connected sensors are running at 3.3V.





Each of the four main sensor ports have an enable pin, which must be set correctly to power the sensor. The enable pins are found here:



The first three pins (pH, ORP and Aux) must be set low to power on the sensor. The last pin (Temp) must be set high to power on the sensor.

Truth table

Pin	State	Sensor Power
pH EN	LOW	ON
ORP EN	LOW	ON
Aux EN	LOW	ON
Temp EN	HIGH	ON

Sensor port 5 (the terminal block) does not have an enable pin and can not be turned off.

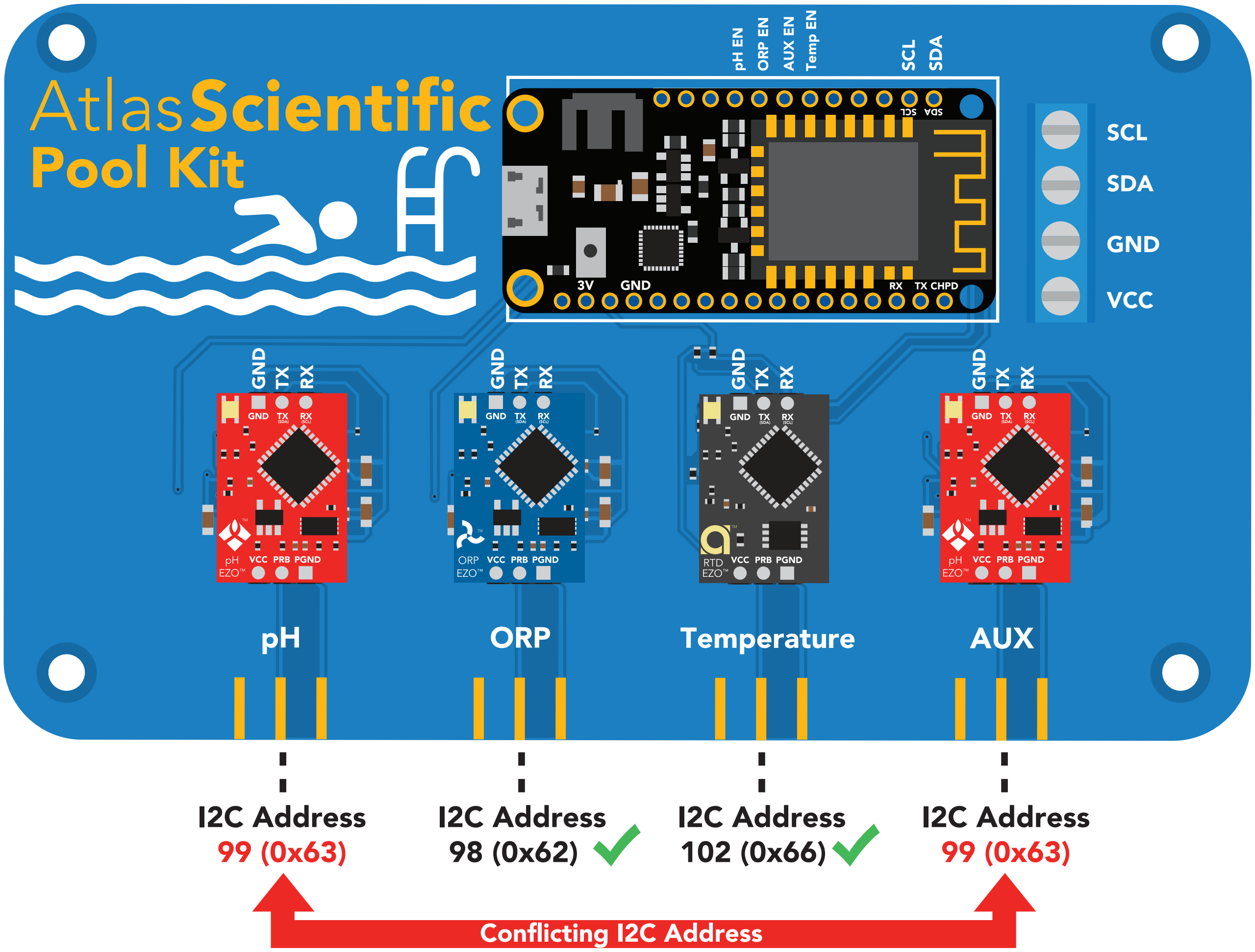
Data protocol

The CPU communicates with all peripheral sensors using the I2C data protocol. All data lines are directly connected to the CPUs I2C port. Using a different data protocol with this circuit board is not possible.

It is important to keep in mind that all Atlas Scientific components default to UART mode. When adding a new Atlas Scientific component to the kit, it must first be put into I2C mode. Refer to the component's datasheet for instructions on how to switch it over.

Adding more of the same sensor or component type

Adding additional components of the same type, such as an additional pH or ORP sensor, is not hard to do. As mentioned above, you must set the device to I2C mode, and you must make sure that its I2C address is not the same as the already existing component.



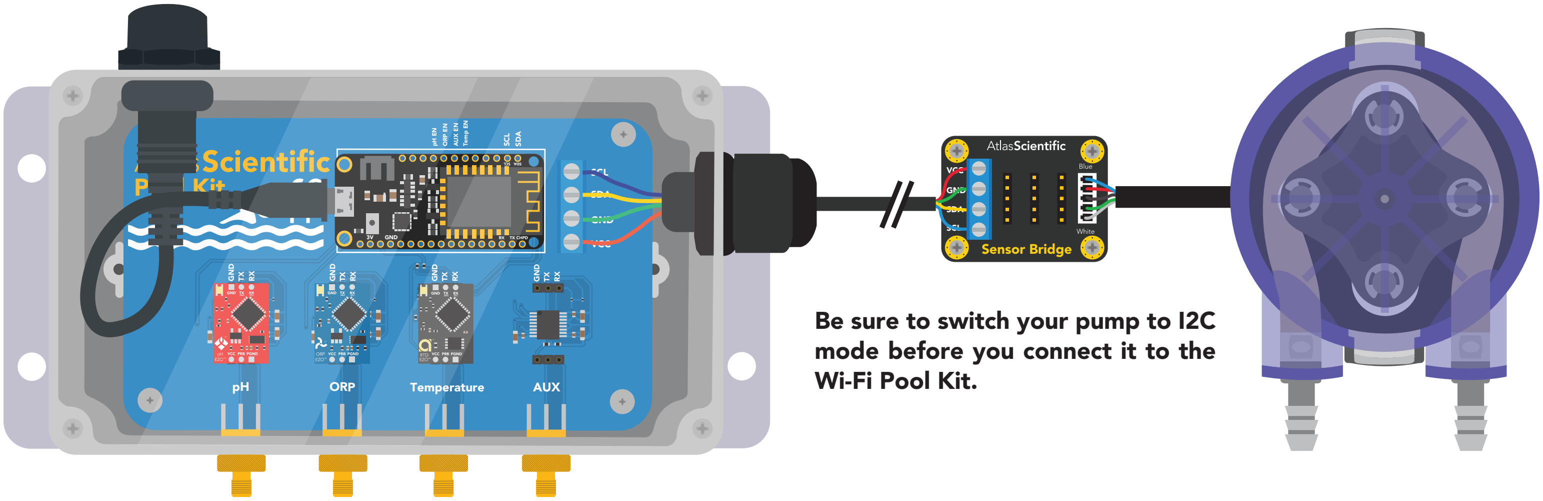
This table lists the default I2C address of components commonly added to this kit.

Device	I2C Address
EZO pH	99 (0x63)
EZO ORP	98 (0x62)
EZO DO	97 (0x61)

Device	I2C Address
EZO EC	100 (0x64)
EZO RTD	102 (0x66)
EZO PMP-L	109 (0x6D)

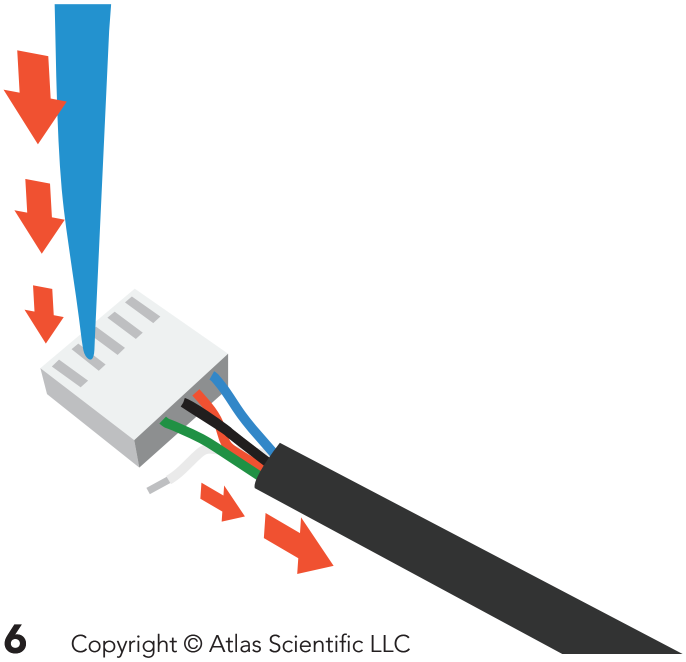
Dosing pump

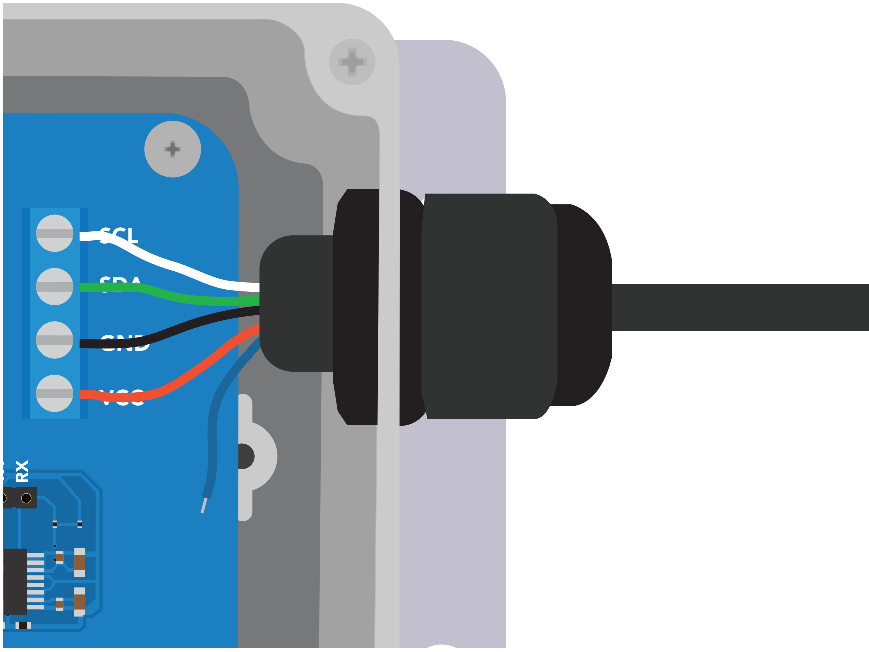
An optional dosing pump can be added to the Wi-Fi Pool Kit. Using both the Large Embedded Dosing Pump and our sensor bridge is the simplest way to add on a dosing pump.



Be sure to switch your pump to I2C mode before you connect it to the Wi-Fi Pool Kit.

You can directly connect an EZO Pump to the Wi-Fi Pool Kit without the sensor bridge however you must remove the data cable connector and manually put the pump into I2C mode.





Uploading sensor data to the cloud

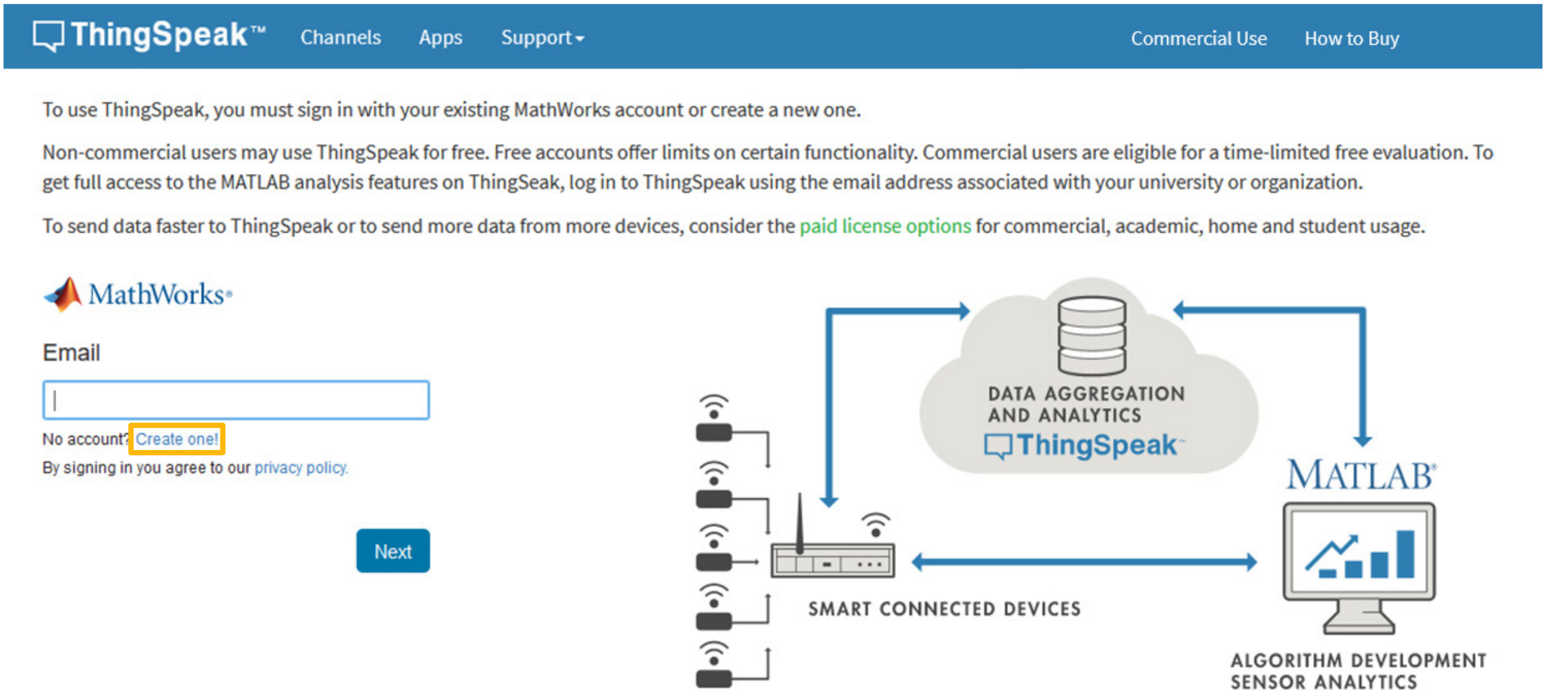
The Atlas-Scientific Wi-Fi Pool Kit has been designed to upload sensor data to ThingSpeak ™, a free, cloud-based data acquisition and visualization platform. You will be required to set up a free account with ThingSpeak ™ to upload and visualize the data. With a free account, you can upload data once every 15 seconds. A paid account lets you upload data once per-second; look here for more info about various ThingSpeak ™ services.

Atlas Scientific has no business relationship with ThingSpeak ™; we just like how it works. If you want to use a different service, modify the device as you see fit.

Setting up your Wi-Fi kit

Step 1 Setup a ThingSpeak Account

Because the sensor data is stored / viewed on ThingSpeak, you will need to setup a ThingSpeak account. Create your ThingSpeak account by clicking HERE.



To use ThingSpeak, you must sign in with your existing MathWorks account or create a new one.

Non-commercial users may use ThingSpeak for free. Free accounts offer limits on certain functionality. Commercial users are eligible for a time-limited free evaluation. To get full access to the MATLAB analysis features on ThingSeak, log in to ThingSpeak using the email address associated with your university or organization.

To send data faster to ThingSpeak or to send more data from more devices, consider the paid license options for commercial, academic, home and student usage.

MathWorks

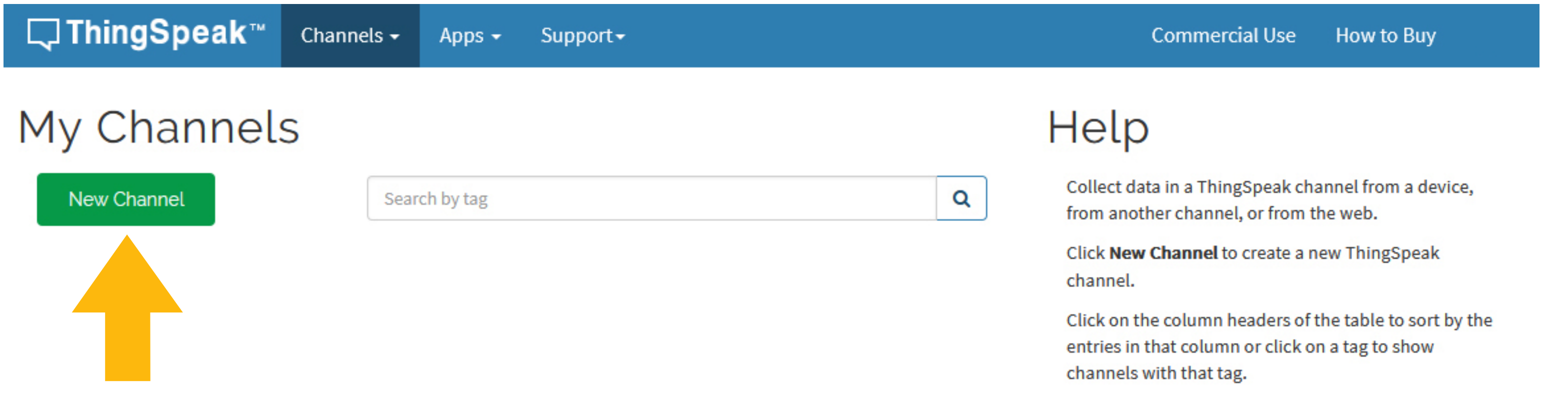
Email

No account? Create one!

By signing in you agree to our privacy policy.

Step 2 Create a Channel

Your data is uploaded to ThingSpeak through a ‘Channel.’ Select **New Channel**



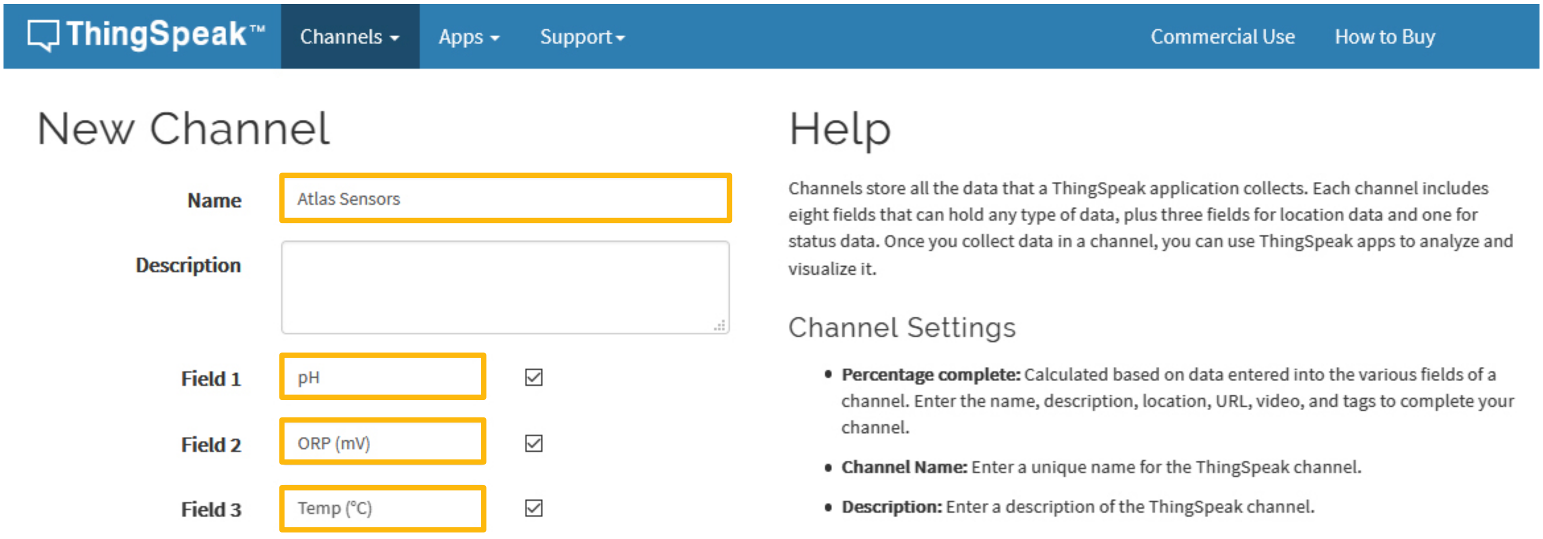
My Channels

Help

Collect data in a ThingSpeak channel from a device, from another channel, or from the web.

Click **New Channel** to create a new ThingSpeak channel.

Click on the column headers of the table to sort by the entries in that column or click on a tag to show channels with that tag.



New Channel

Name	Atlas Sensors	
Description		
Field 1	pH	☑
Field 2	ORP (mV)	☑
Field 3	Temp (°C)	☑

Help

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

Channel Settings

- **Percentage complete:** Calculated based on data entered into the various fields of a channel. Enter the name, description, location, URL, video, and tags to complete your channel.
- **Channel Name:** Enter a unique name for the ThingSpeak channel.
- **Description:** Enter a description of the ThingSpeak channel.

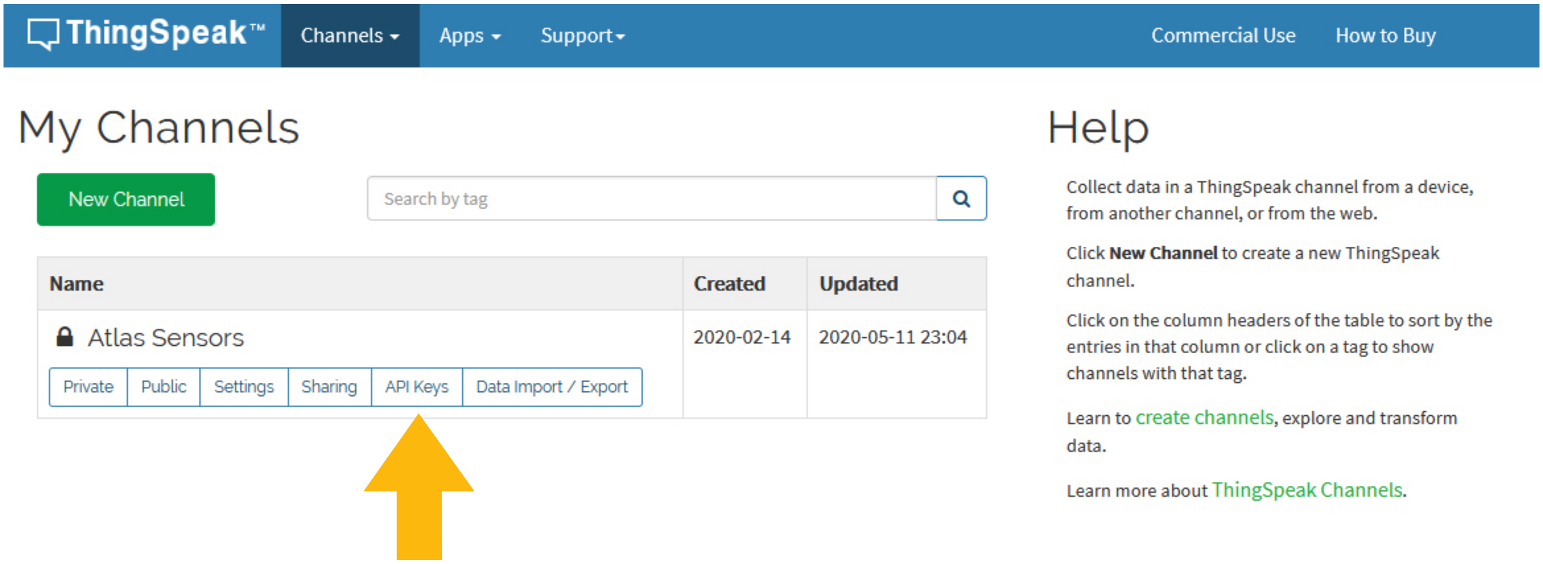
Fill out the highlighted boxes. (Be sure to click on the checkboxes to enable **field 2** and **3**)
For reference, this is what we entered.

Name	**Atlas Sensors**
Field 1	**pH**
Field 2	**ORP (mV)**
Field 3	**Temp (°C)**

Scroll to the bottom of the page and click **Save Channel**.

Step 3 Get ThingSpeak API keys

After you saved your channel settings, you will be redirected to your channel page.
Click on **API keys**.



My Channels

Name	Created	Updated
Atlas Sensors	2020-02-14	2020-05-11 23:04

Help

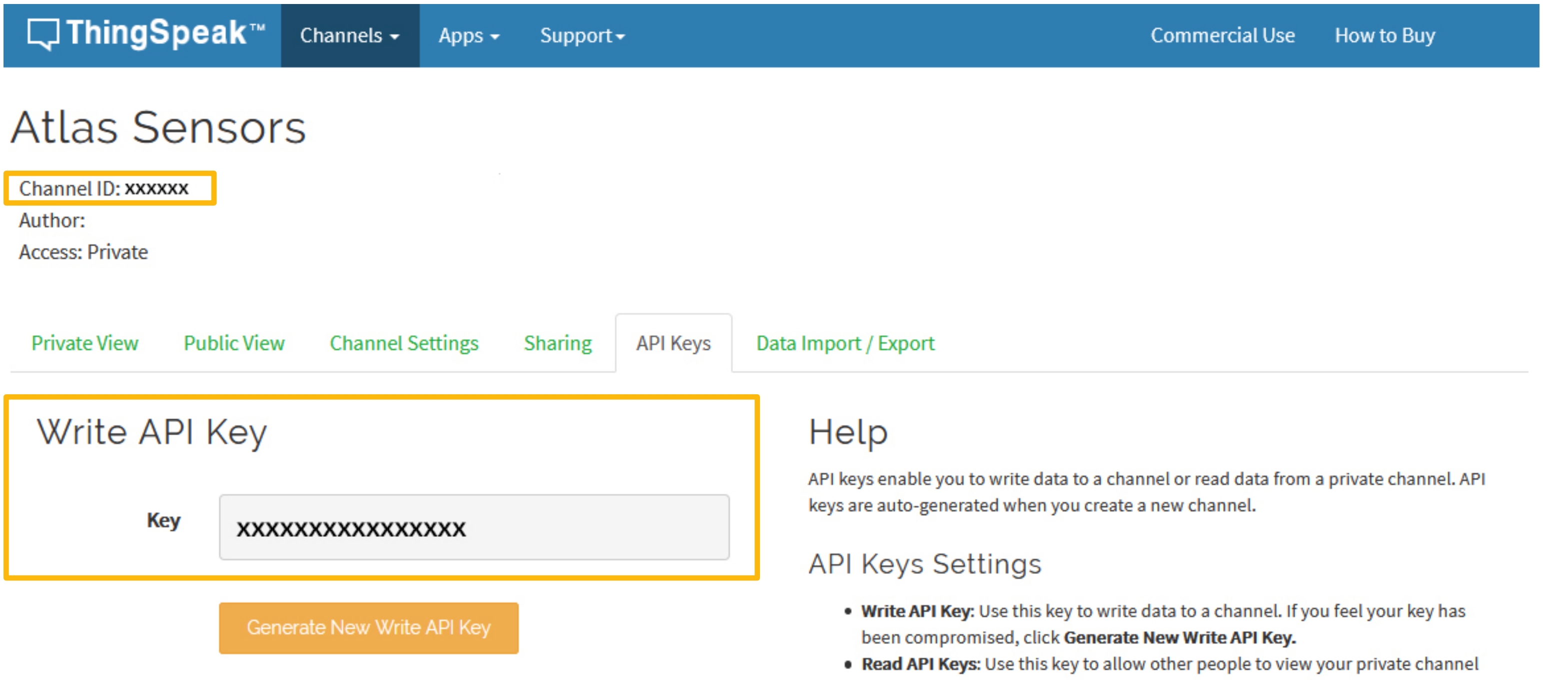
Collect data in a ThingSpeak channel from a device, from another channel, or from the web.

Click **New Channel** to create a new ThingSpeak channel.

Click on the column headers of the table to sort by the entries in that column or click on a tag to show channels with that tag.

Learn to create channels, explore and transform data.

Learn more about ThingSpeak Channels.



Atlas Sensors

Channel ID: **xxxxxx**
Author:
Access: Private

Write API Key

Key XXXXXXXXXXXXXXXX

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

API Keys Settings

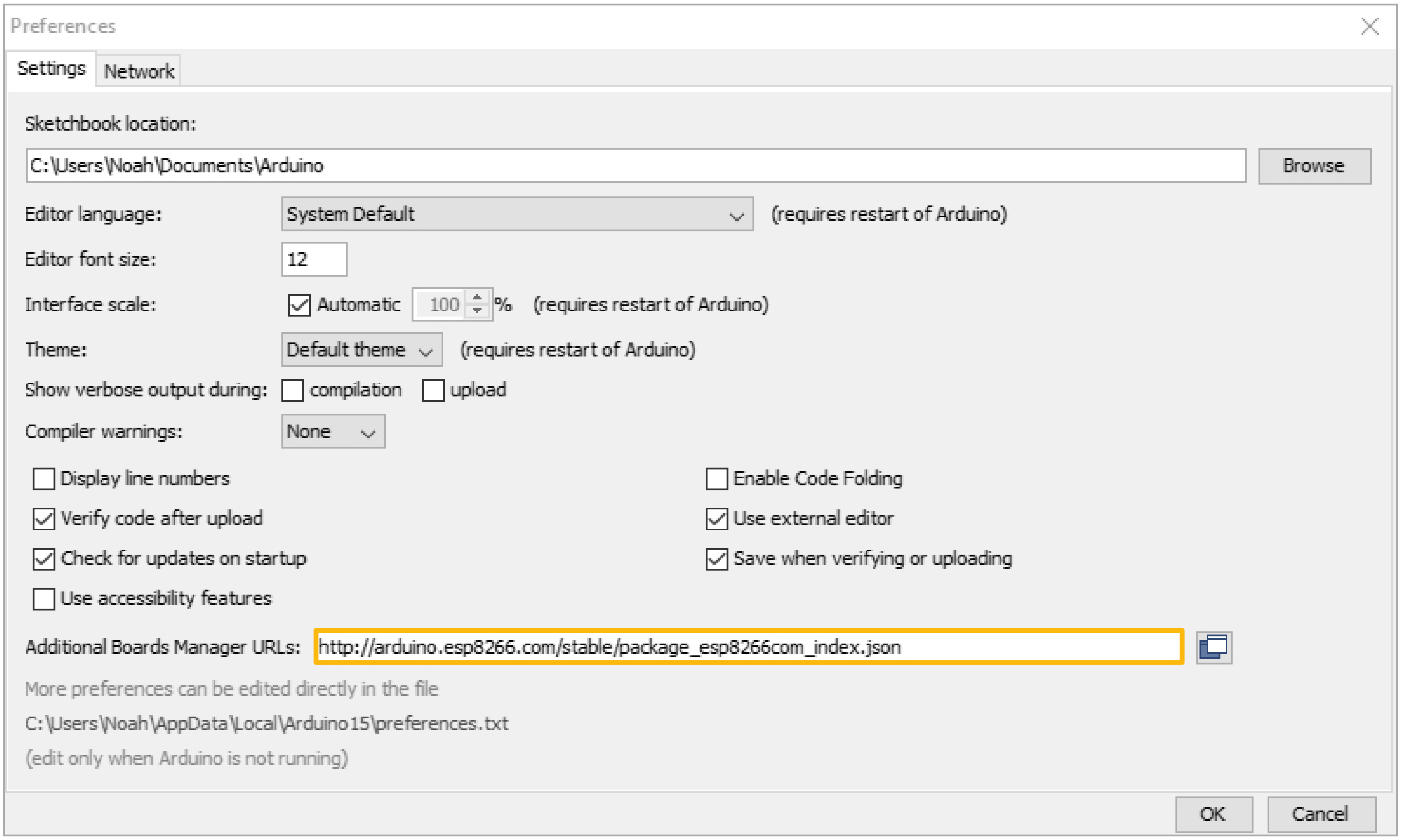
- **Write API Key**: Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API Key.**
- **Read API Keys**: Use this key to allow other people to view your private channel

Be sure to save your **Channel ID** and **Write API Key** we are going to need these, in the next few steps.

Step 4 Make sure your Arduino IDE libraries are up to date

A Make sure you have the correct path for the Esp8266 Library

In the IDE, go to **File > Preferences**
Locate the **Additional Boards Manager URLS** text box.



Sketchbook location: C:\Users\Noah\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

Interface scale: ☑ Automatic 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: □ compilation □ upload

Compiler warnings: None

□ Display line numbers
☑ Verify code after upload
☑ Check for updates on startup
□ Use accessibility features
□ Enable Code Folding
☑ Use external editor
☑ Save when verifying or uploading

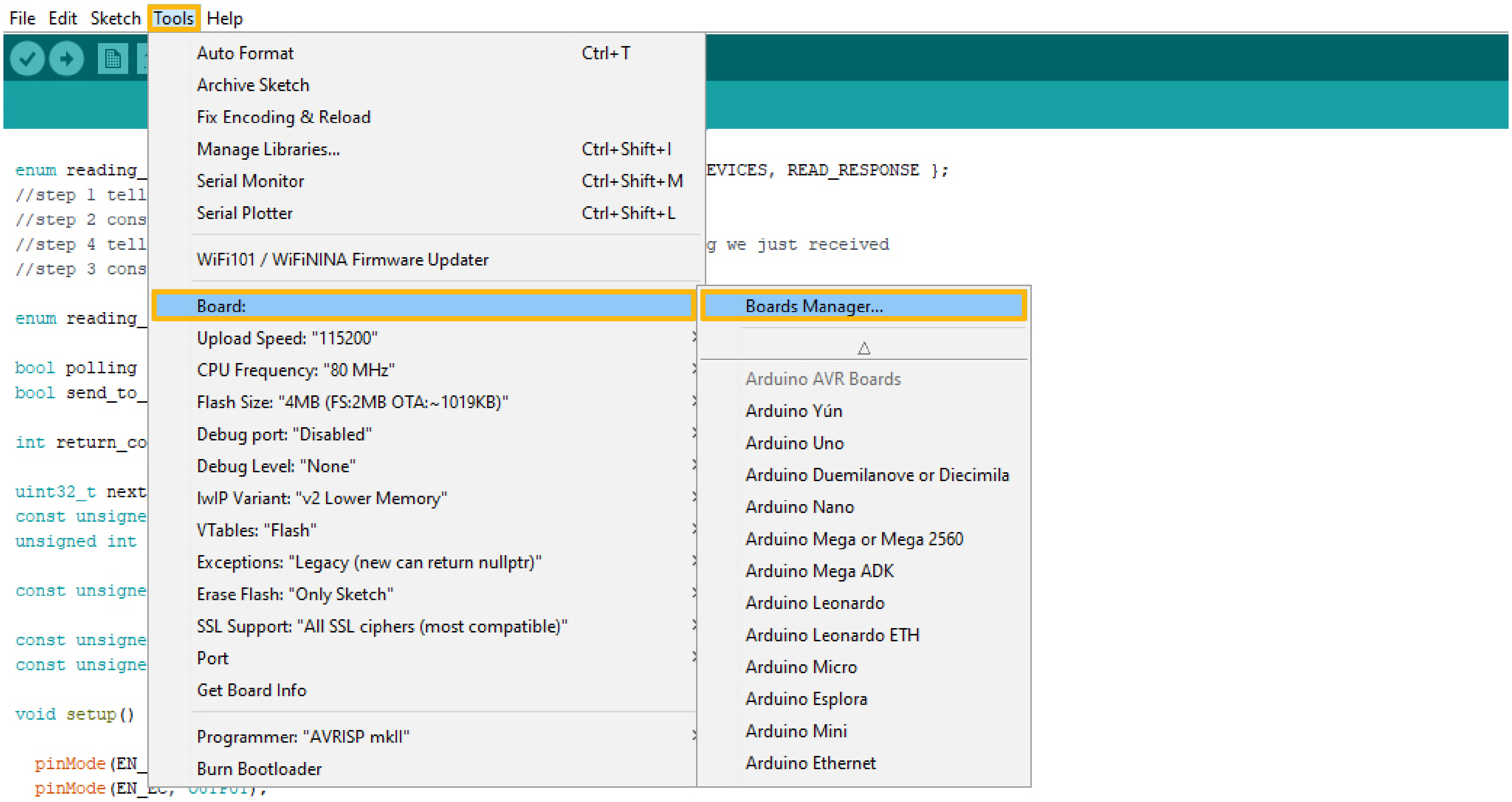
Additional Boards Manager URLs: http://arduino.esp8266.com/stable/package_esp8266com_index.json

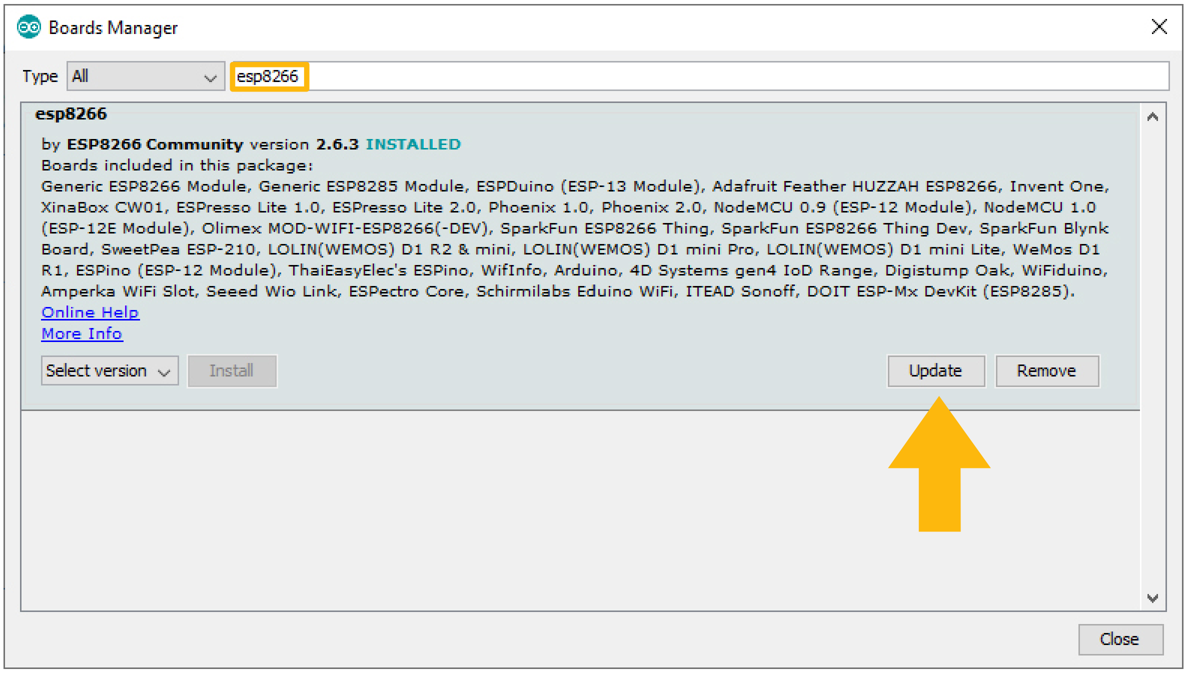
More preferences can be edited directly in the file
C:\Users\Noah\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

Make sure this URL is in the textbox
http://arduino.esp8266.com/stable/package_esp8266com_index.json
Click **OK**.

B Update the esp8266 board

In the IDE, go to **Tools > Board > Boards Manager**





esp8266

by **ESP8266 Community** version **2.6.3** **INSTALLED**
Boards included in this package:
Generic ESP8266 Module, Generic ESP8285 Module, ESPDuino (ESP-13 Module), Adafruit Feather HUZZAH ESP8266, Invent One, XinaBox CW01, ESPresso Lite 1.0, ESPresso Lite 2.0, Phoenix 1.0, Phoenix 2.0, NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Olimex MOD-WIFI-ESP8266(-DEV), SparkFun ESP8266 Thing, SparkFun ESP8266 Thing Dev, SparkFun Blynk Board, SweetPea ESP-210, LOLIN(WEMOS) D1 R2 & mini, LOLIN(WEMOS) D1 mini Pro, LOLIN(WEMOS) D1 mini Lite, WeMos D1 R1, ESPino (ESP-12 Module), ThaiEasyElec's ESPino, WifInfo, Arduino, 4D Systems gen4 IoD Range, Digistump Oak, WiFiduino, Amperka WiFi Slot, Seeed Wio Link, ESPectro Core, Schirmilabs Eduino WiFi, ITEAD Sonoff, DOIT ESP-Mx DevKit (ESP8285).
Online Help
More Info

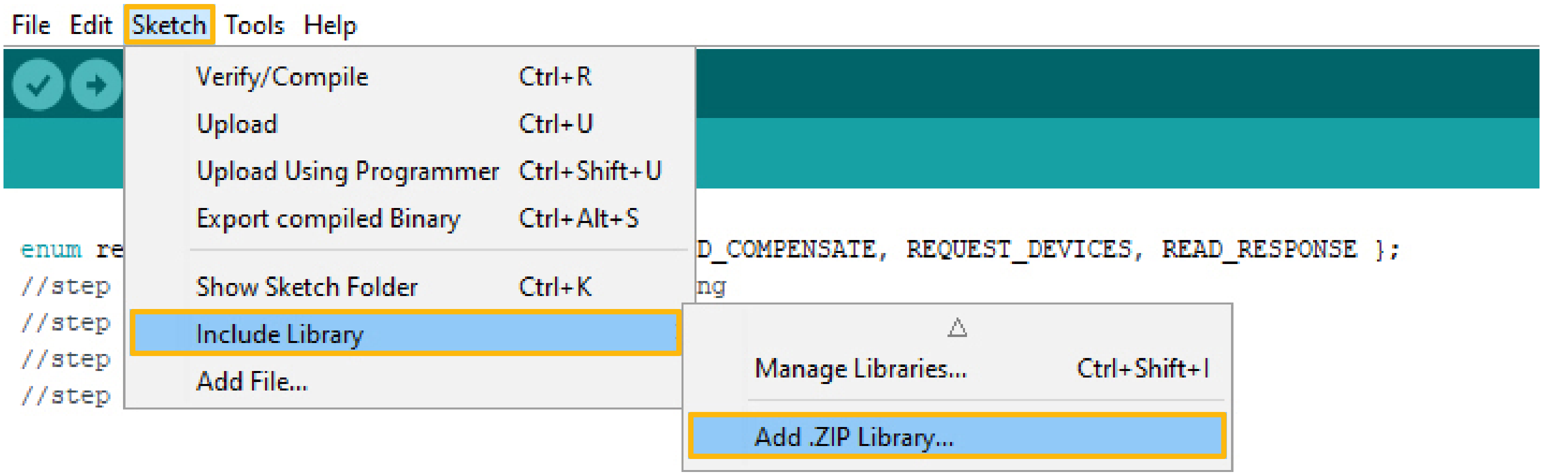
In the search bar of the Boards Manager, lookup **esp8266.**
Update to the most recent version if you don't already have it.

(Version 2.6.3 in not the most recent version)

C Download the ThingSpeak library for Arduino

Click HERE to download the latest version of the ThingSpeak library.

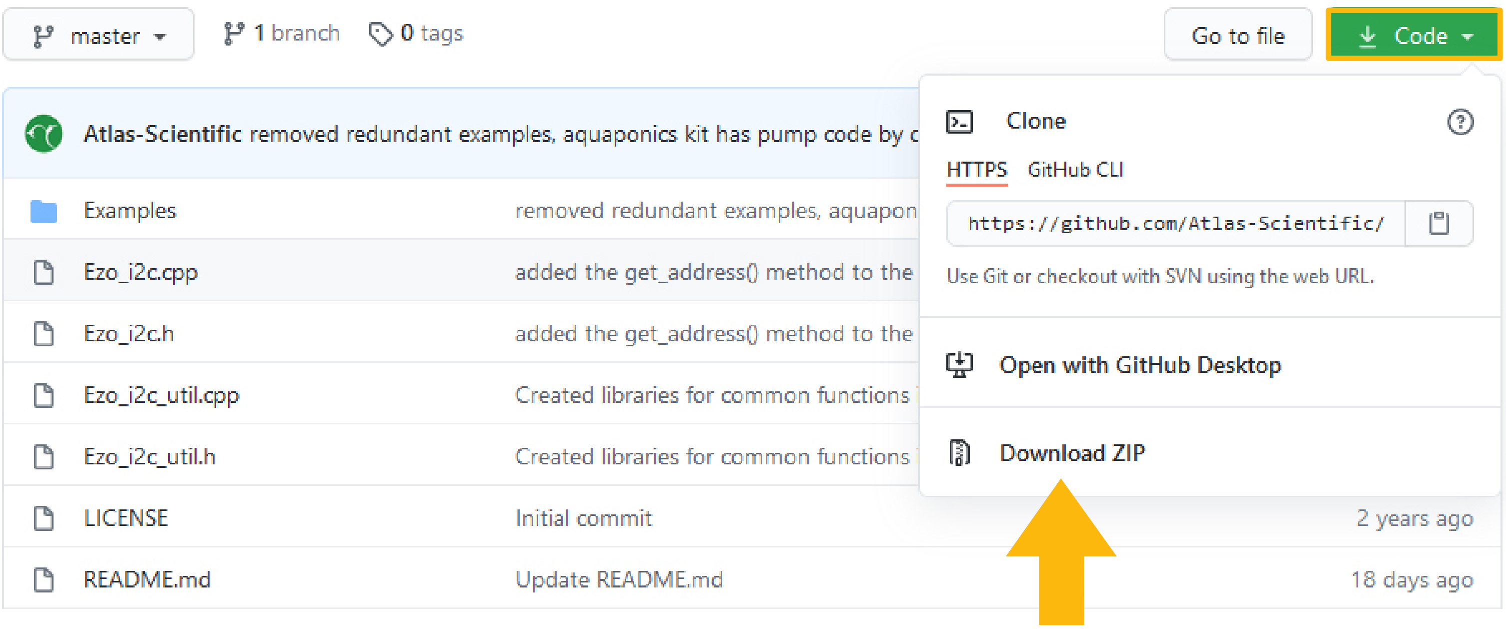
Don't unzip it!
Import the .ZIP file into your Arduino IDE.
To import the .ZIP file go to **Sketch > Include Library > Add .ZIP Library**


File Edit Sketch Tools Help
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
Manage Libraries... Ctrl+Shift+I
Add .ZIP Library...

```
enum re                                       D_COMPENSATE, REQUEST_DEVICES, READ_RESPONSE };
//step                                        ng
//step
//step
//step
```

D Add the EZO I2C Library

To download the Ezo_I2c library file, click HERE.



https://github.com/Atlas-Scientific/

Use Git or checkout with SVN using the web URL.

Atlas-Scientific removed redundant examples, aquaponics kit has pump code by c

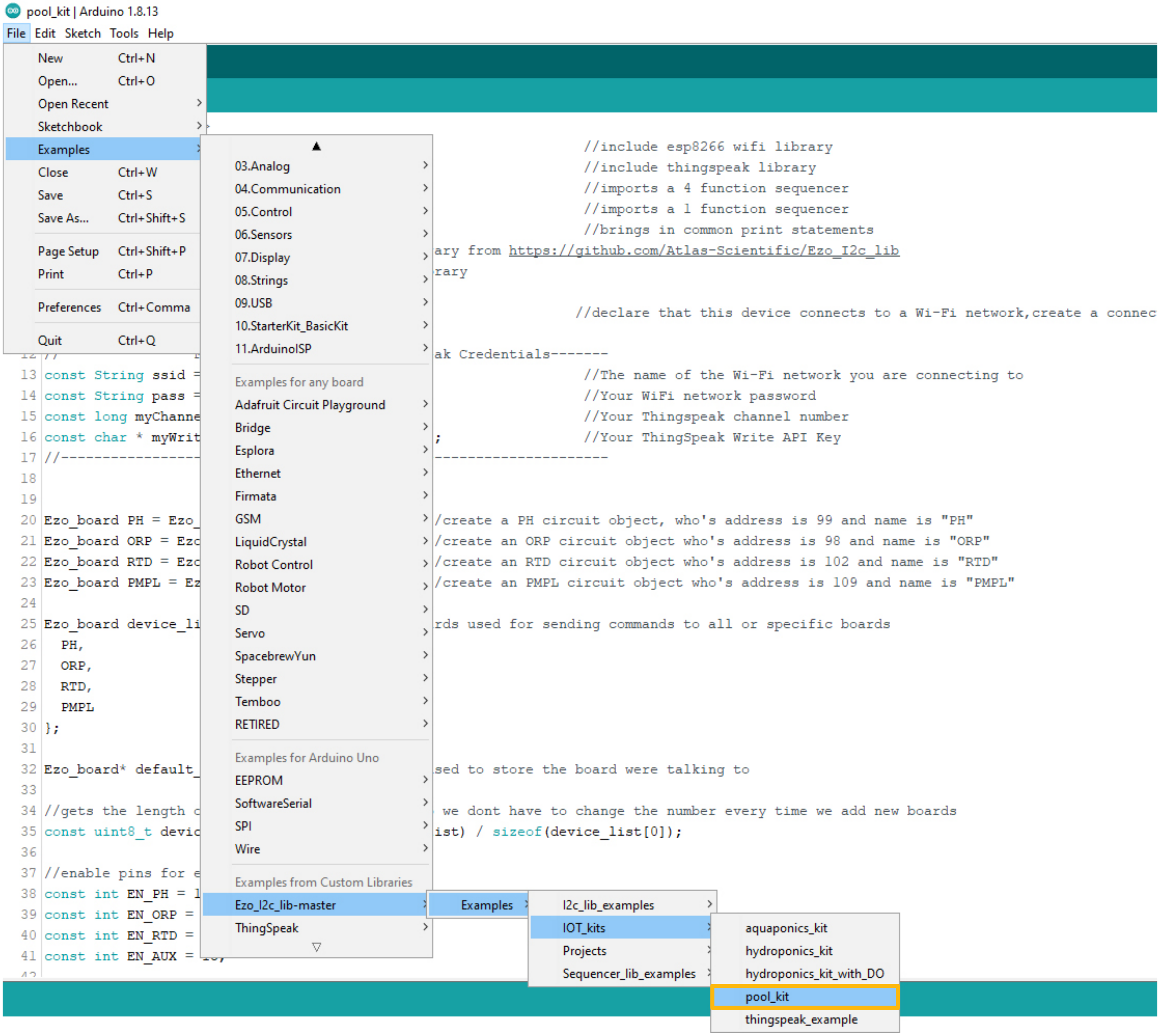
Examples	removed redundant examples, aquapon	
Ezo_i2c.cpp	added the get_address() method to the	
Ezo_i2c.h	added the get_address() method to the	
Ezo_i2c_util.cpp	Created libraries for common functions	
Ezo_i2c_util.h	Created libraries for common functions	
LICENSE	Initial commit	2 years ago
README.md	Update README.md	18 days ago

Don't unzip it!
Import the .ZIP file to your Arduino IDE.
To import the .ZIP file go to **Sketch > Include Library > Add .ZIP Library**

Step 5 Flash the Pool kit with the correct code

A Select, open and adjust the code you want to use for your Wi-Fi Kit

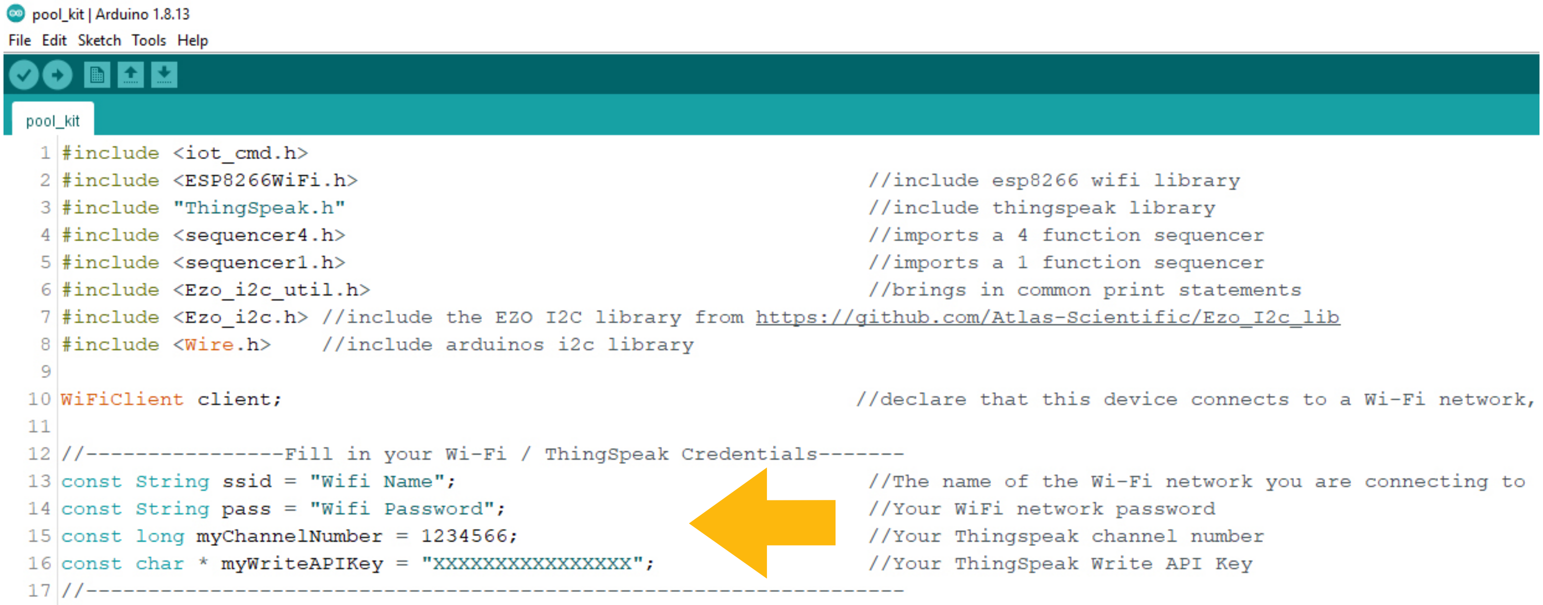
File> Examples> EZO_I2C_lib-master> Examples> IOT_kits> pool_kit



```
                                             //include esp8266 wifi library
                                             //include thingspeak library
                                             //imports a 4 function sequencer
                                             //imports a 1 function sequencer
                                             //brings in common print statements
ary from https://github.com/Atlas-Scientific/Ezo_I2c_lib
rary

                                            //declare that this device connects to a Wi-Fi network,create a connec

//                                    ak Credentials-------
const String ssid =                          //The name of the Wi-Fi network you are connecting to
const String pass =                          //Your WiFi network password
const long myChanne                          //Your Thingspeak channel number
const char * myWrit  ;                       //Your ThingSpeak Write API Key
//-----------------  ---------------------


Ezo_board PH = Ezo_  /create a PH circuit object, who's address is 99 and name is "PH"
Ezo_board ORP = Ezo  /create an ORP circuit object who's address is 98 and name is "ORP"
Ezo_board RTD = Ezo  /create an RTD circuit object who's address is 102 and name is "RTD"
Ezo_board PMPL = Ez  /create an PMPL circuit object who's address is 109 and name is "PMPL"

Ezo_board device_li  rds used for sending commands to all or specific boards
  PH,
  ORP,
  RTD,
  PMPL
};

Ezo_board* default_  sed to store the board were talking to

//gets the length o   we dont have to change the number every time we add new boards
const uint8_t devic  ist) / sizeof(device_list[0]);

//enable pins for e
const int EN_PH = 1
const int EN_ORP =
const int EN_RTD =
const int EN_AUX = 1
```

B Fill in your Wi-Fi / ThingSpeak credentials

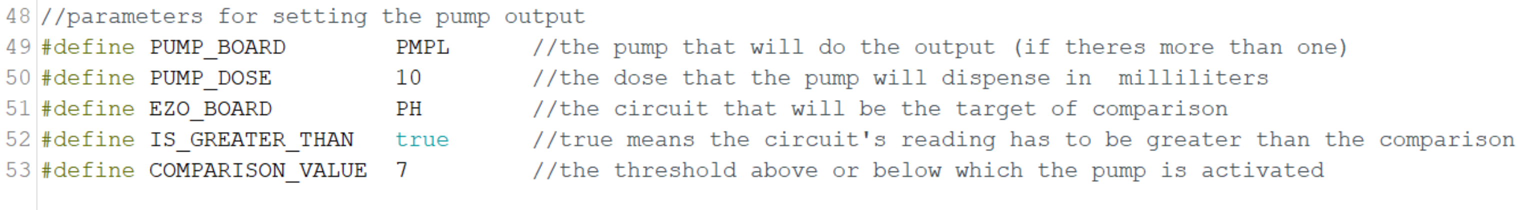
Fill in your Wi-Fi name and Password, along with the Channel ID and Write API Key to the code. *(see step 3)*



```
#include <iot_cmd.h>
#include <ESP8266WiFi.h>                                                //include esp8266 wifi library
#include "ThingSpeak.h"                                                 //include thingspeak library
#include <sequencer4.h>                                                 //imports a 4 function sequencer
#include <sequencer1.h>                                                 //imports a 1 function sequencer
#include <Ezo_i2c_util.h>                                               //brings in common print statements
#include <Ezo_i2c.h> //include the EZO I2C library from https://github.com/Atlas-Scientific/Ezo_I2c_lib
#include <Wire.h>    //include arduinos i2c library

WiFiClient client;                                                     //declare that this device connects to a Wi-Fi network,

//----------------Fill in your Wi-Fi / ThingSpeak Credentials-------
const String ssid = "Wifi Name";                                        //The name of the Wi-Fi network you are connecting to
const String pass = "Wifi Password";                                    //Your WiFi network password
const long myChannelNumber = 1234566;                                   //Your Thingspeak channel number
const char * myWriteAPIKey = "XXXXXXXXXXXXXXXX";                        //Your ThingSpeak Write API Key
//------------------------------------------------------------------
```

C Setting up your pump

If you do not have a pump attached, you can just skip this part. The code is rather self explanatory. You set what parameters will trigger the pump to engage.

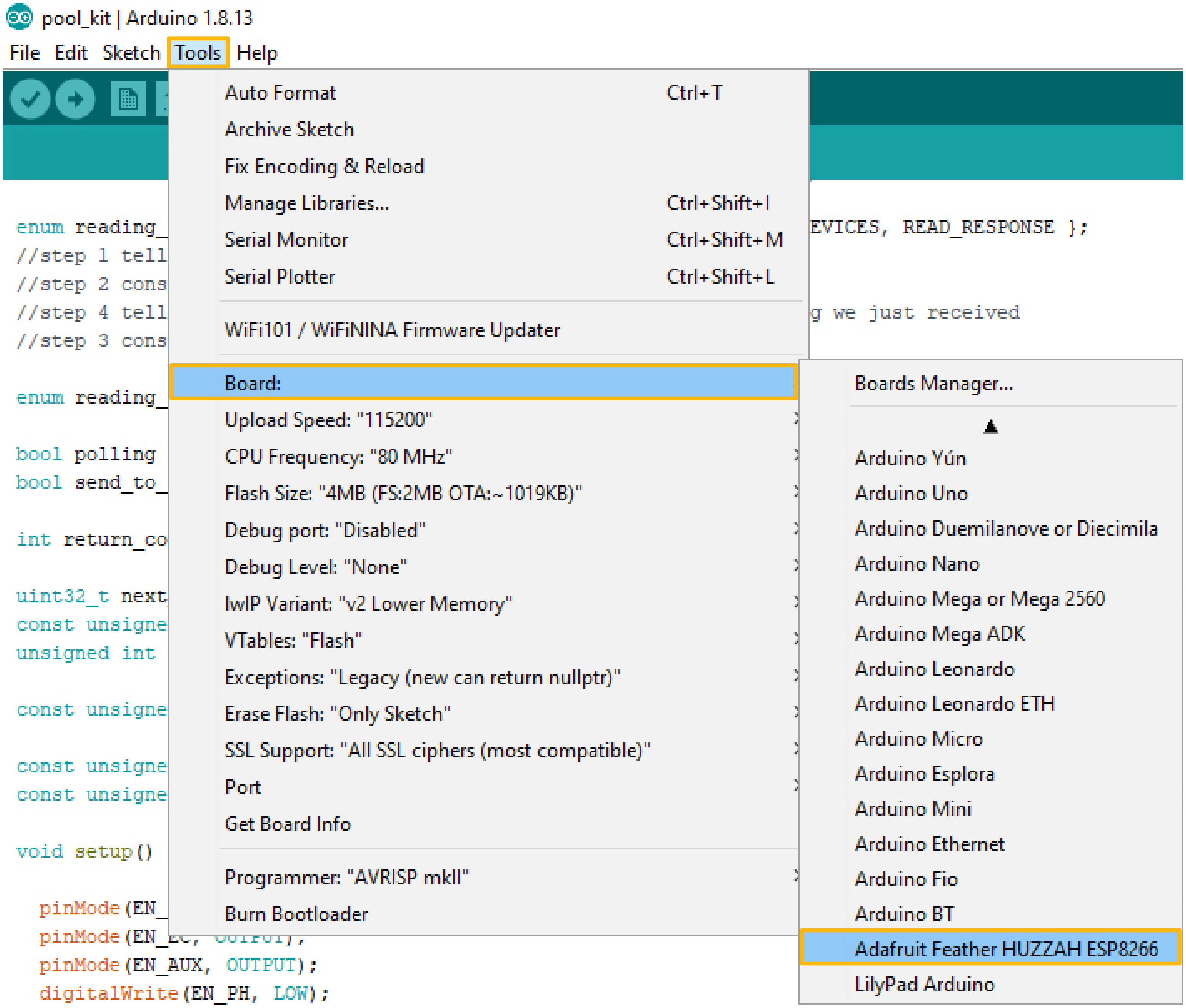


```
//parameters for setting the pump output
#define PUMP_BOARD        PMPL      //the pump that will do the output (if theres more than one)
#define PUMP_DOSE         10        //the dose that the pump will dispense in  milliliters
#define EZO_BOARD         PH        //the circuit that will be the target of comparison
#define IS_GREATER_THAN   true      //true means the circuit's reading has to be greater than the comparison
#define COMPARISON_VALUE  7         //the threshold above or below which the pump is activated
```

Step 6 Setting up the HUZZAH board

A Set the target CPU to flash

Tools> Board> Adafruit Feather HUZZAH ESP8266



Upload Speed: "115200"

CPU Frequency: "80 MHz"

Flash Size: "4MB (FS:2MB OTA:~1019KB)"

Debug port: "Disabled"

Debug Level: "None"

lwIP Variant: "v2 Lower Memory"

VTables: "Flash"

Exceptions: "Legacy (new can return nullptr)"

Erase Flash: "Only Sketch"

SSL Support: "All SSL ciphers (most compatible)"

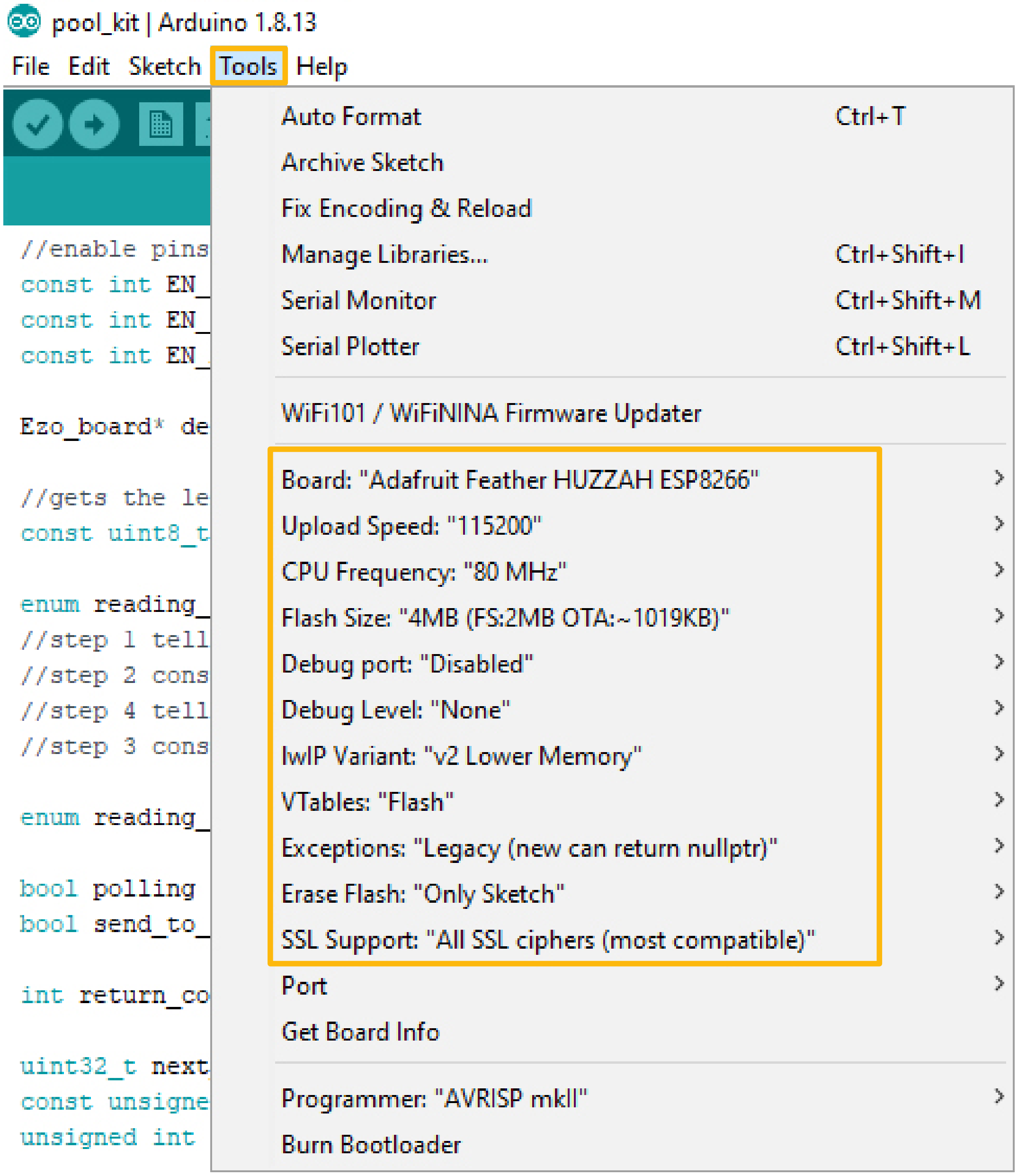
Programmer: "AVRISP mkII"

```
enum reading_                    EVICES, READ_RESPONSE };
//step 1 tell
//step 2 cons
//step 4 tell                    g we just received
//step 3 cons

enum reading_

bool polling
bool send_to_

int return_co

uint32_t next
const unsigne
unsigned int

const unsigne

const unsigne
const unsigne

void setup()

  pinMode(EN_
  pinMode(EN_EC, OUTPUT);
  pinMode(EN_AUX, OUTPUT);
  digitalWrite(EN_PH, LOW);
```

B Adjust CPU Settings

Make sure the CPU settings on the Adafruit Feather HUZZAH ESP8266 are correct.
To adjust the CPU settings, click **Tools**.

For reference, this is what Atlas Scientific set the CPU settings to.
(your options may not be exactly the same, just try and match them as closely as possible.)

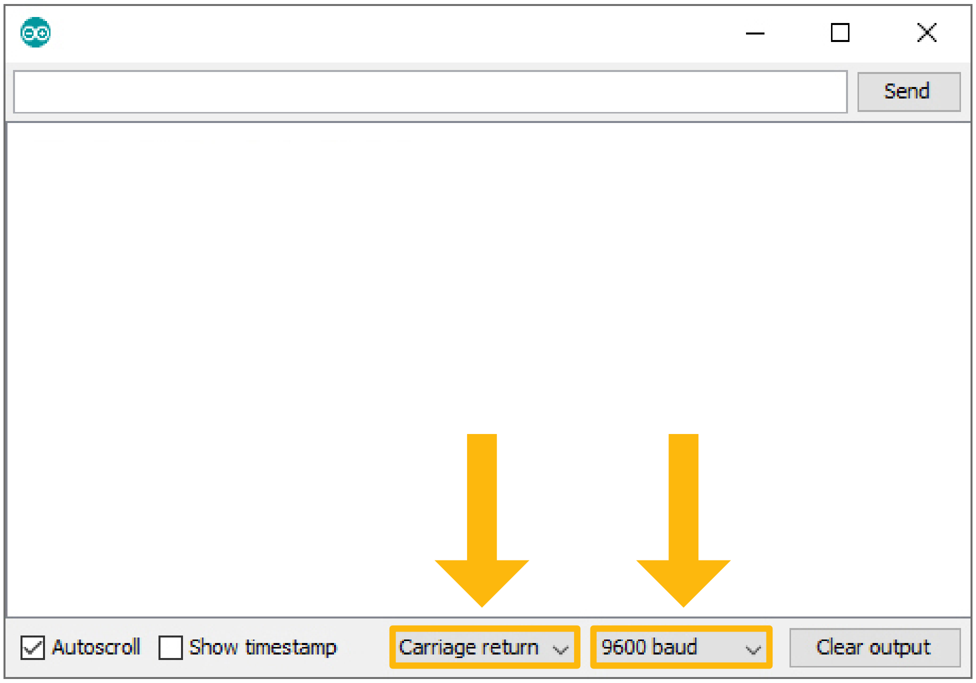

pool_kit | Arduino 1.8.13
File
Edit
Sketch
Tools
Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader

Board: "Adafruit Feather HUZZAH ESP8266"
Upload Speed: "115200"
CPU Frequency: "80 MHz"
Flash Size: "4MB (FS:2MB OTA:~1019KB)"
Debug port: "Disabled"
Debug Level: "None"
IwIP Variant: "v2 Lower Memory"
VTables: "Flash"
Exceptions: "Legacy (new can return nullptr)"
Erase Flash: "Only Sketch"
SSL Support: "All SSL ciphers (most compatible)"

```
//enable pins
const int EN_
const int EN_
const int EN_

Ezo_board* de

//gets the le
const uint8_t

enum reading_
//step 1 tell
//step 2 cons
//step 4 tell
//step 3 cons

enum reading_

bool polling
bool send_to_

int return_co

uint32_t next
const unsigne
unsigned int
```

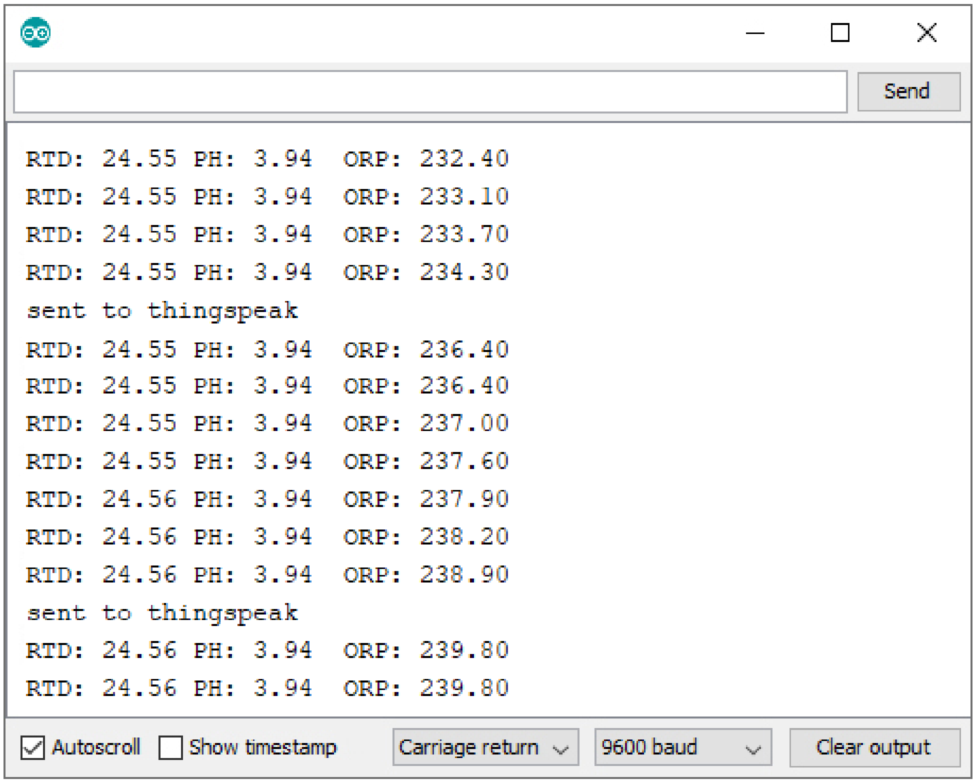
Step 7 See the readings

Open your Arduino serial monitor.
(You must have the serial monitor set to the com port from the Adafruit Feather HUZZAH)



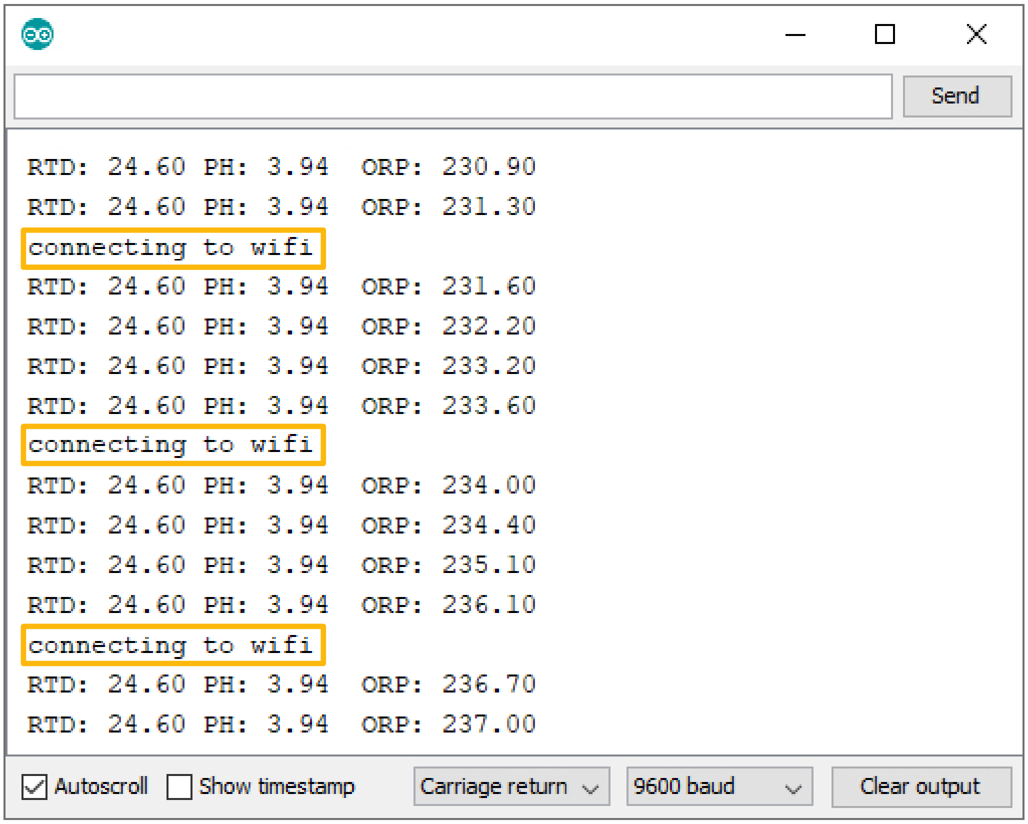
Set to **carriage return** and **9600 baud.**

The Wi-Fi Pool Kit will always attempt to connect to ThingSpeak on bootup.



```
RTD: 24.55 PH: 3.94  ORP: 232.40
RTD: 24.55 PH: 3.94  ORP: 233.10
RTD: 24.55 PH: 3.94  ORP: 233.70
RTD: 24.55 PH: 3.94  ORP: 234.30
sent to thingspeak
RTD: 24.55 PH: 3.94  ORP: 236.40
RTD: 24.55 PH: 3.94  ORP: 236.40
RTD: 24.55 PH: 3.94  ORP: 237.00
RTD: 24.55 PH: 3.94  ORP: 237.60
RTD: 24.56 PH: 3.94  ORP: 237.90
RTD: 24.56 PH: 3.94  ORP: 238.20
RTD: 24.56 PH: 3.94  ORP: 238.90
sent to thingspeak
RTD: 24.56 PH: 3.94  ORP: 239.80
RTD: 24.56 PH: 3.94  ORP: 239.80
```

If it cannot connect to your Wi-Fi you will see this:



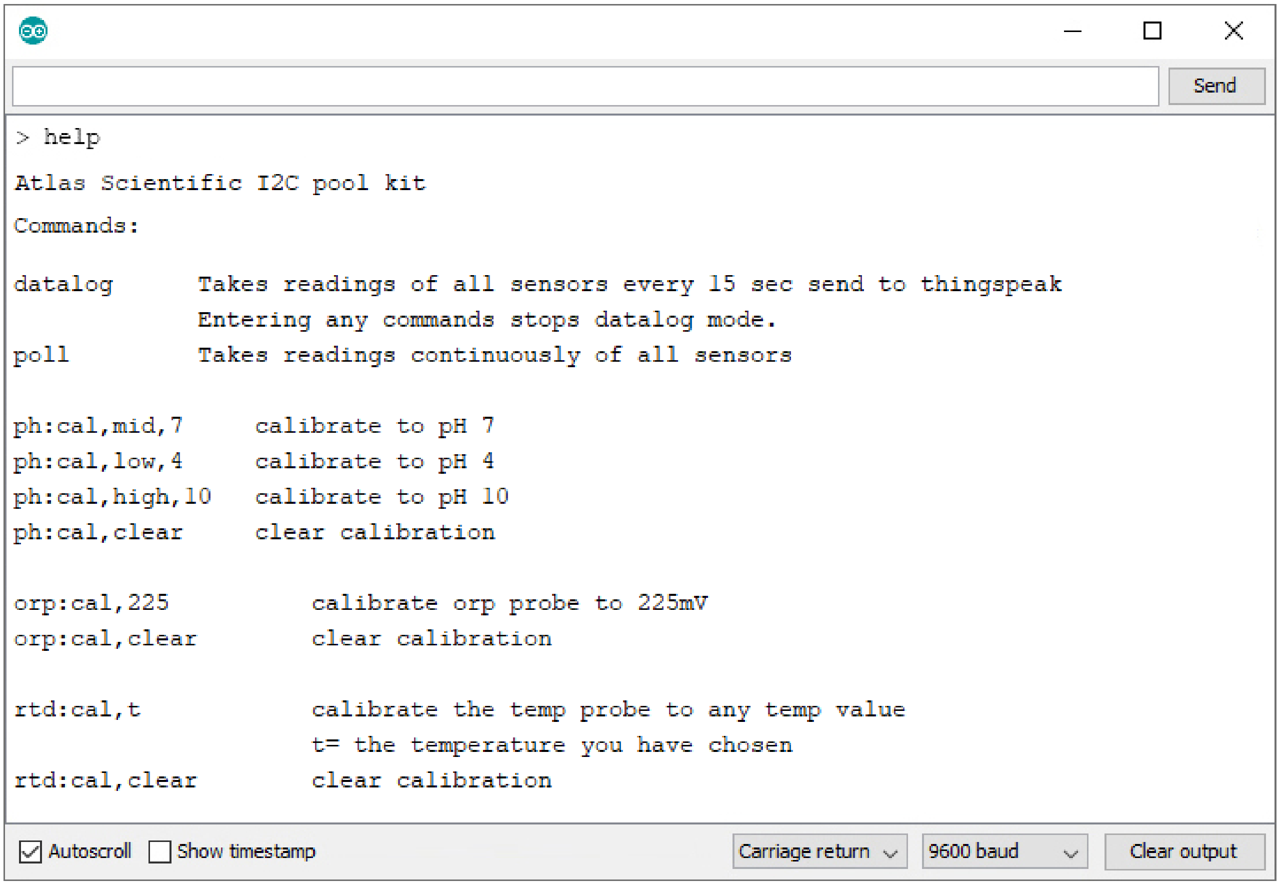
```
RTD: 24.60 PH: 3.94  ORP: 230.90
RTD: 24.60 PH: 3.94  ORP: 231.30
connecting to wifi
RTD: 24.60 PH: 3.94  ORP: 231.60
RTD: 24.60 PH: 3.94  ORP: 232.20
RTD: 24.60 PH: 3.94  ORP: 233.20
RTD: 24.60 PH: 3.94  ORP: 233.60
connecting to wifi
RTD: 24.60 PH: 3.94  ORP: 234.00
RTD: 24.60 PH: 3.94  ORP: 234.40
RTD: 24.60 PH: 3.94  ORP: 235.10
RTD: 24.60 PH: 3.94  ORP: 236.10
connecting to wifi
RTD: 24.60 PH: 3.94  ORP: 236.70
RTD: 24.60 PH: 3.94  ORP: 237.00
```

☑ Autoscroll □ Show timestamp Carriage return 9600 baud

Entering the **poll** command will stop the Wi-Fi Pool Kit from uploading the readings to thingspeak, while you debug your Wifi problems.

Step 8 Sensor Calibration

Atlas Scientific created a list of calibration commands that are built into the library. Type in **help** to see a list of commands.



```
> help
Atlas Scientific I2C pool kit
Commands:

datalog      Takes readings of all sensors every 15 sec send to thingspeak
             Entering any commands stops datalog mode.
poll         Takes readings continuously of all sensors

ph:cal,mid,7     calibrate to pH 7
ph:cal,low,4     calibrate to pH 4
ph:cal,high,10   calibrate to pH 10
ph:cal,clear     clear calibration

orp:cal,225          calibrate orp probe to 225mV
orp:cal,clear        clear calibration

rtd:cal,t            calibrate the temp probe to any temp value
                     t= the temperature you have chosen
rtd:cal,clear        clear calibration
```

☑ Autoscroll □ Show timestamp | Carriage return | 9600 baud

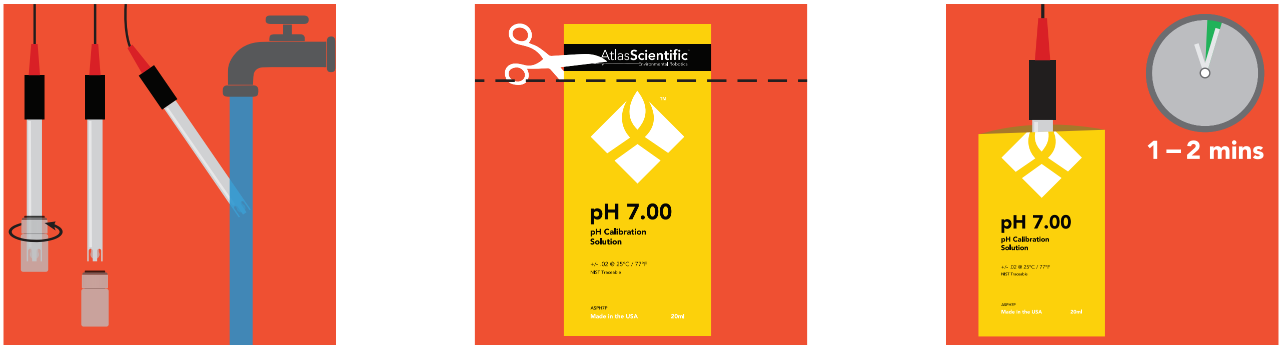
A The poll command

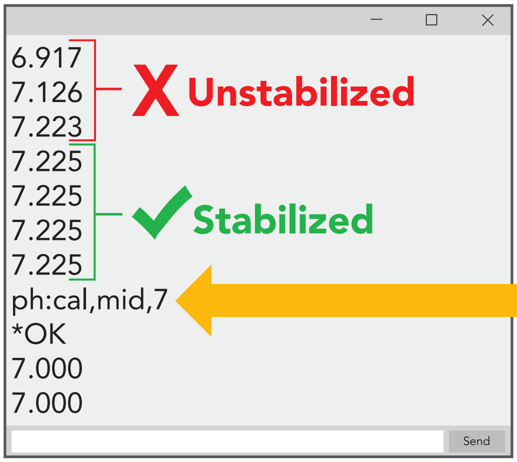
Send the command **poll**; This will let you see the readings once per second and it will stop uploading to ThingSpeak while you calibrate.

B Calibrate pH

When calibrating pH, you must always calibrate to pH 7 first.

Remove the soaker bottle and rinse off the pH probe. Remove the top of the pH 7.00 calibration solution pouch. Place the pH probe inside the pouch and let the probe sit in the calibration solution until the readings stabilize. This will take about 1 – 2 mins.





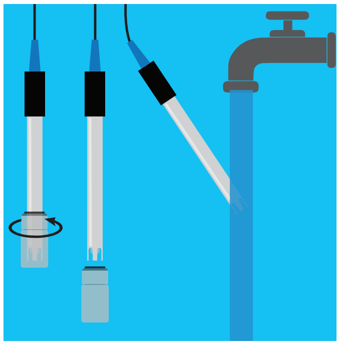
```
6.917
7.126
7.223
7.225
7.225
7.225
7.225
ph:cal,mid,7
*OK
7.000
7.000
```

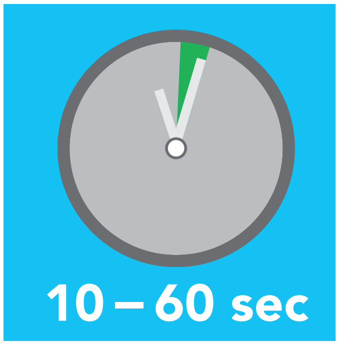
Once the readings have stabilized, issue the Mid point calibration command. **ph:cal,mid,7**

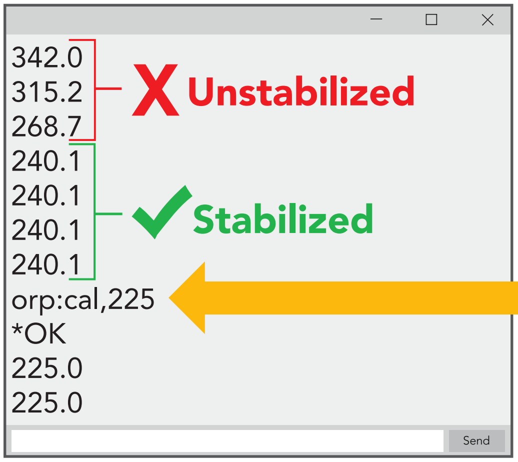
Rinse off the probe and repeat this process for both **pH 4.00** and **pH 10.00**.

C Calibrate ORP

Rinse off the probe, and insert it directly into the 225mV calibration solution, and watch the readings. Wait for the ORP readings to stabilize. This will take about 10 – 60 seconds.







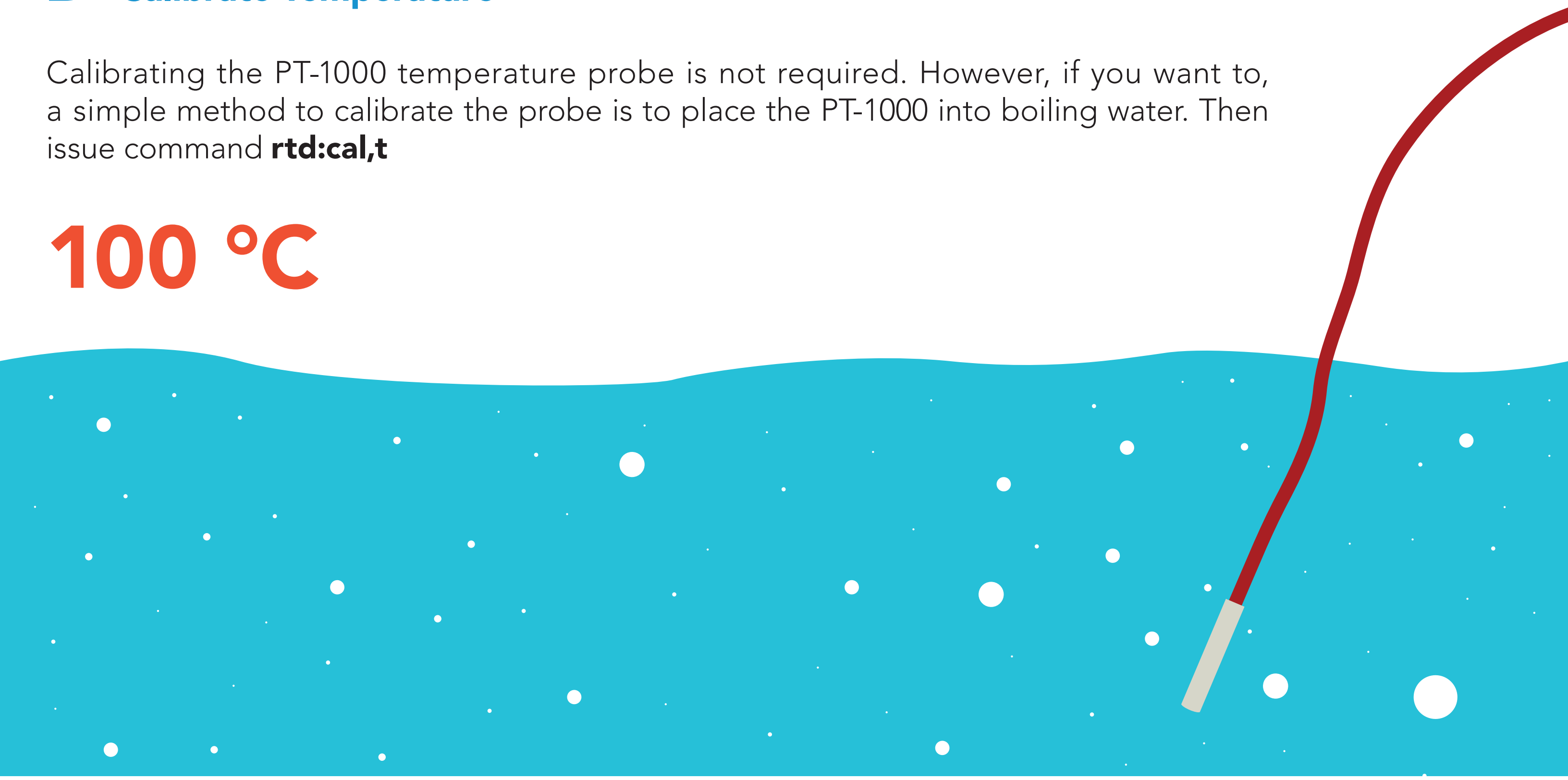
```
342.0
315.2
268.7
240.1
240.1
240.1
240.1
orp:cal,225
*OK
225.0
225.0
```

Once the readings have stabilized, issue the calibration command. In this case **orp:cal,225**

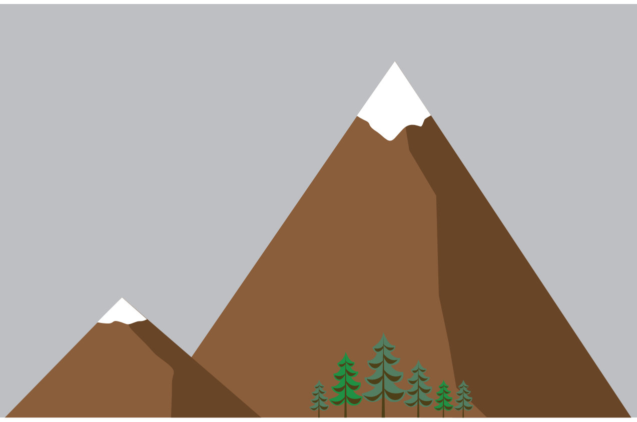
D Calibrate Temperature

Calibrating the PT-1000 temperature probe is not required. However, if you want to, a simple method to calibrate the probe is to place the PT-1000 into boiling water. Then issue command **rtd:cal,t**

100 °C



Elevation in meters	Boiling point
305	98.9 °C
229	99.2 °C
152	99.5 °C
76	99.7 °C
0	100 °C
-76	100.3 °C
-152	100.5 °C



Calibration Complete

Step 9 Almost done!

Once you are finished with calibration, issue the **datalog** command to resume taking a reading every 15 seconds and uploading it to thingspeak.

To see the data on your phone, download the ThingSpeak app.

Setup Complete!