



# Workshop: Build Your First IoT Project!

# About Makerden

We're an Educational Makerspace where we combine the necessary tools, equipment, and instruction for hands-on learning through building things.

Members of our group vary both widely and wildly in terms of skill level and age range (10 to 68).



# About Makerden

## **Events organized at our Educational Makerspace:**

**[DIY]** These events include show-and-tells by our members followed by a work-on-your-project(s) and ask-for-help sessions.

**[Workshop]** An instructor leads a hands-on workshop for learning or polishing a practical skill useful for Makers (including schematic-drawing, PCB design, 3D-Printing, soldering).

**[Class]** An instructor leads a formal lecture featuring instructor-led activities that allow participants to learn, develop, or refine a technical skill.

# About ACROBOTIC

## **Makerden's Educational Electronics partner**

Small, bootstrapped Open-Source electronics startup dedicated to the design of hardware and software products for use in education, DIY, hobby, arts, science, and more!

- Online store: [\*\*https://acrobotic.com\*\*](https://acrobotic.com)
- Develops online tutorials: [\*\*http://learn.acrobotic.com\*\*](http://learn.acrobotic.com)
- Supplies all the electronic components for this class!
- Helps Makerden members take their ideas from concept to implementation

# Downloading this presentation

MakerdenIO/First\_IoT\_Proj X

GitHub, Inc. [US] https://github.com/MakerdenIO/First\_IoT\_Project

 [MakerdenIO / First\\_IoT\\_Project](#)

[Unwatch](#) 1 [Star](#) 0 [Fork](#) 0

**Navigate to:**

**[https://github.com/makerdenio/First\\_IoT\\_Project](https://github.com/makerdenio/First_IoT_Project)**

Build an Internet-connected Weather Station using the ESP8266 Development Board! — [Edit](#)

13 commits 1 branch 0 releases 1 contributor

Branch: master [First\\_IoT\\_Project](#) / +

themakerbro updating Latest commit 1a88405 2 minutes ago

iot\_weather\_station updating widgets section 5 days ago

presentation updating 9 minutes ago

LICENSE.txt updating 2 minutes ago

README.md updating 2 minutes ago

README.md

## First\_IoT\_Project

Build an Internet-connected Weather Station using the ESP8266 Development Board!

### Description

For more details, check out the tutorial page at:

Code Issues 0 Pull requests 0 Wiki Pulse Graphs Settings

HTTPS clone URL <https://github.com>

You can clone with [HTTPS](#), [SSH](#), or [Subversion](#). [?](#)

[Clone in Desktop](#) [Download ZIP](#)

# **Intro**

Overview of IoT and ESP8266

Project description

# What is *the* Internet?

The Internet is a global system of interconnected computer networks that communicate with one another to link **billions** of devices worldwide.



*Facebook's Map of World 'Friendship'*

# What devices are found on *the Internet*?

“interconnected computer networks...”

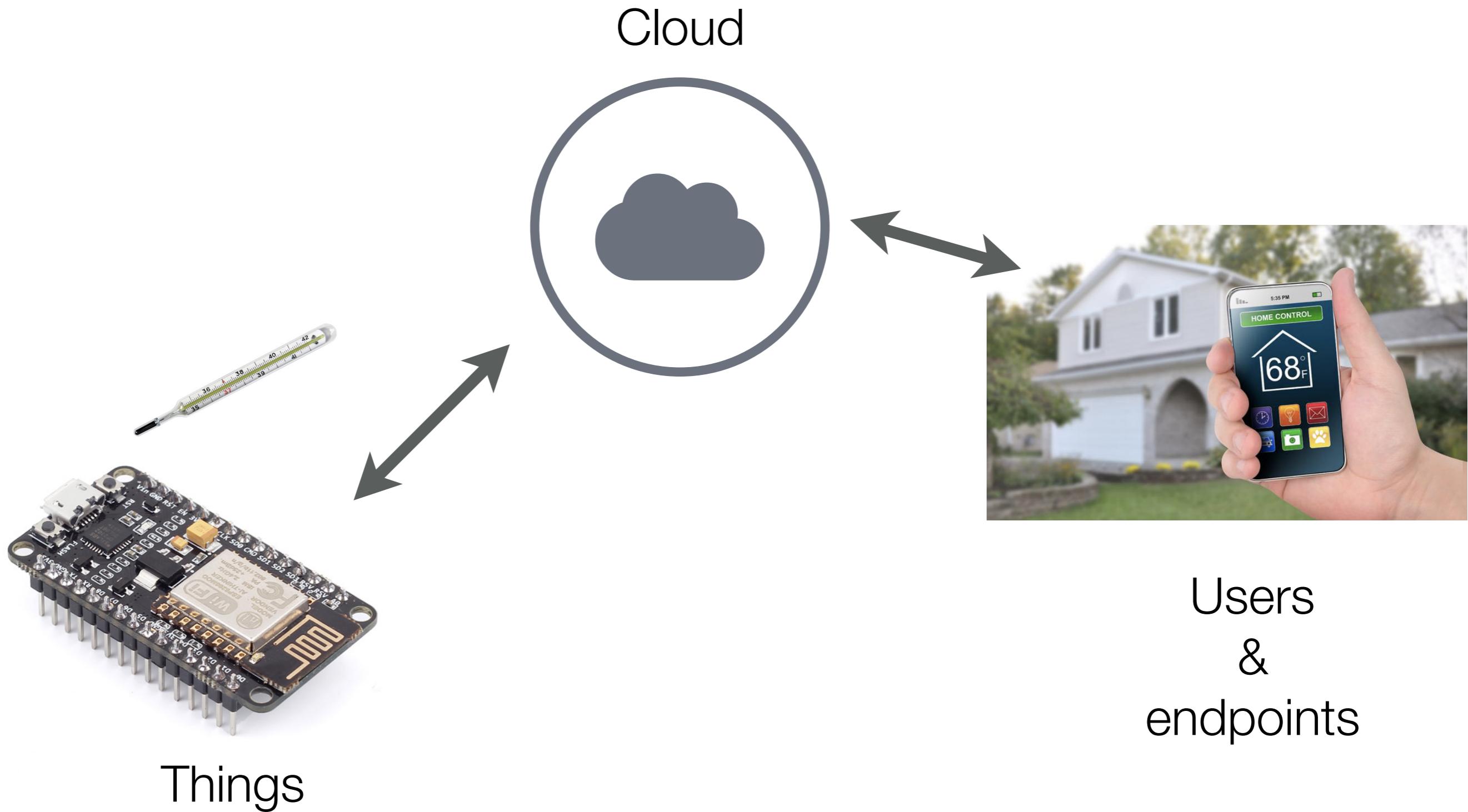


# What is the goal *the* IoT?

Extending the internet to include “everyday objects”.

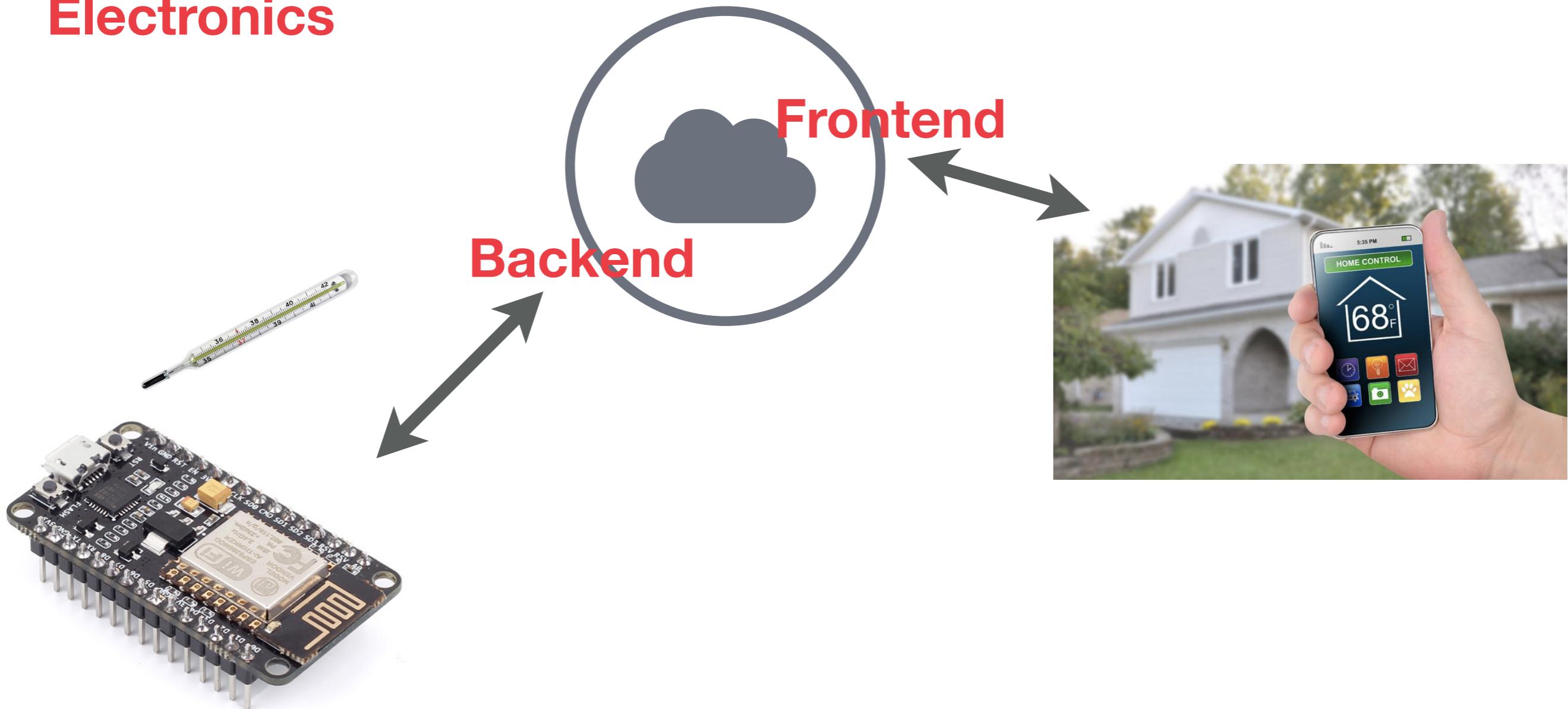


# System Overview



# System Overview

## Electronics

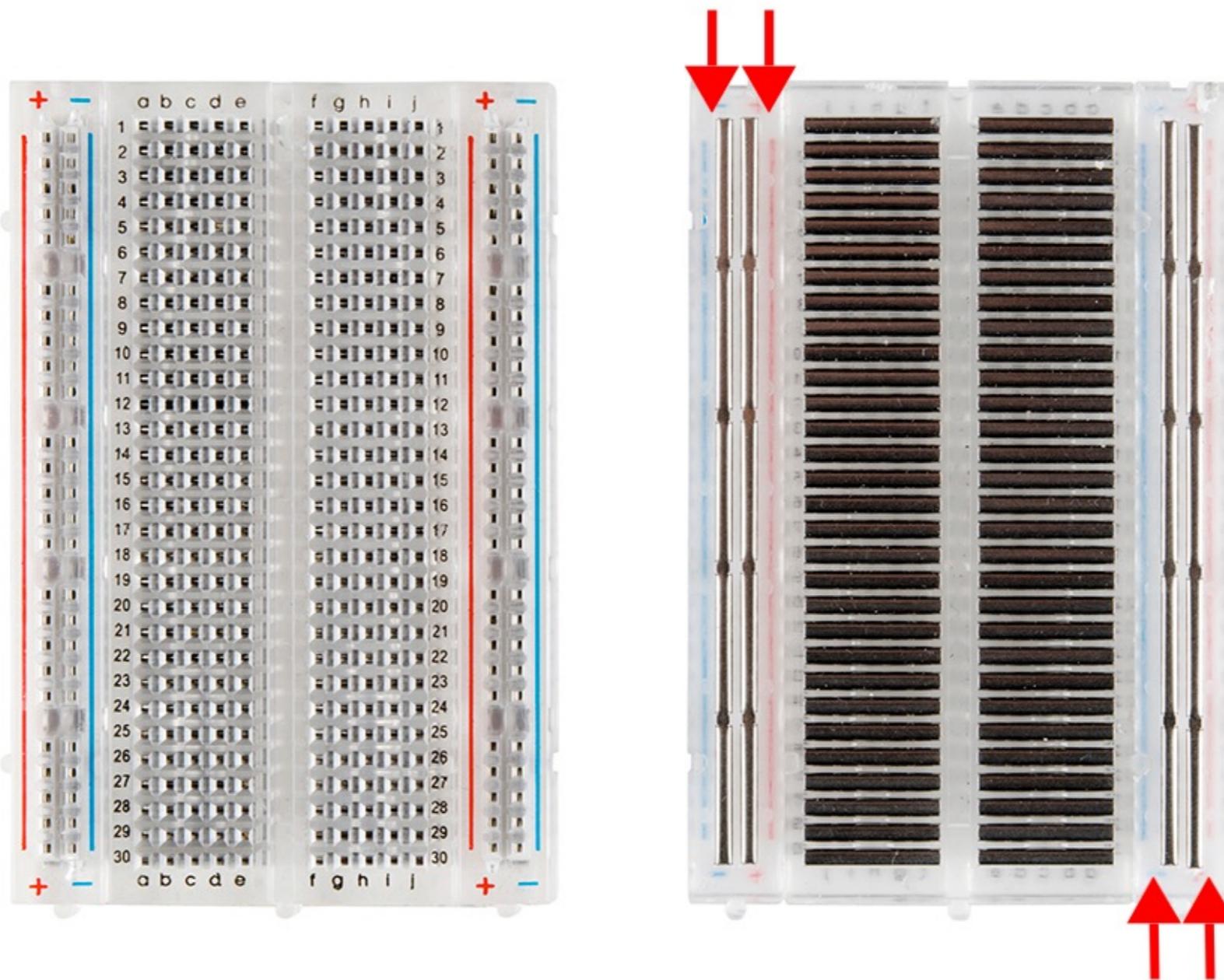


# **Fundamental Concepts**

Hardware prototyping

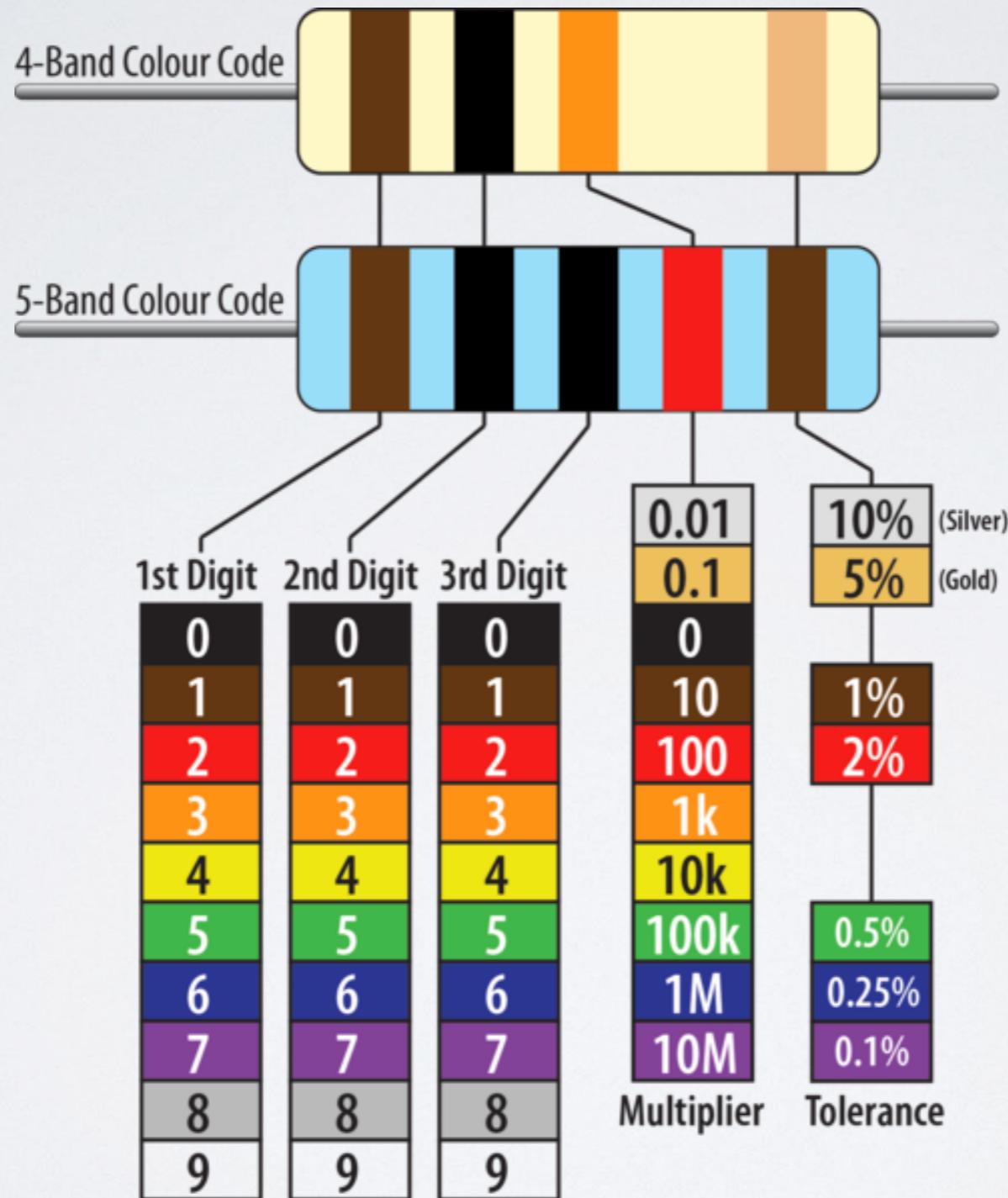
# Electronic Fundamentals

## How the solderless breadboard works:



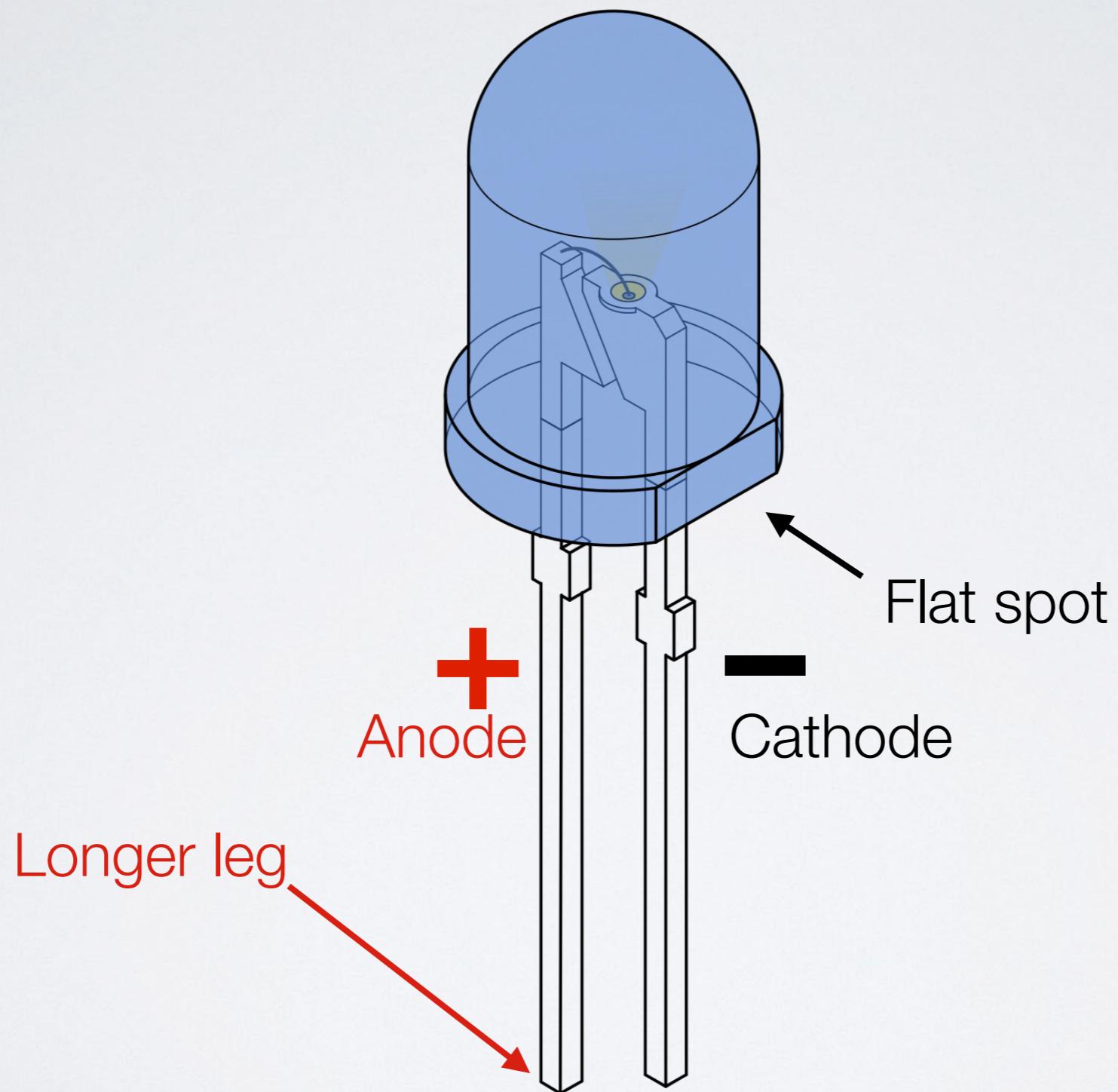
# Electronic Fundamentals

## Reading Resistor Codes:



# Electronic Fundamentals

## Reading LED polarity:



# Electronic Fundamentals

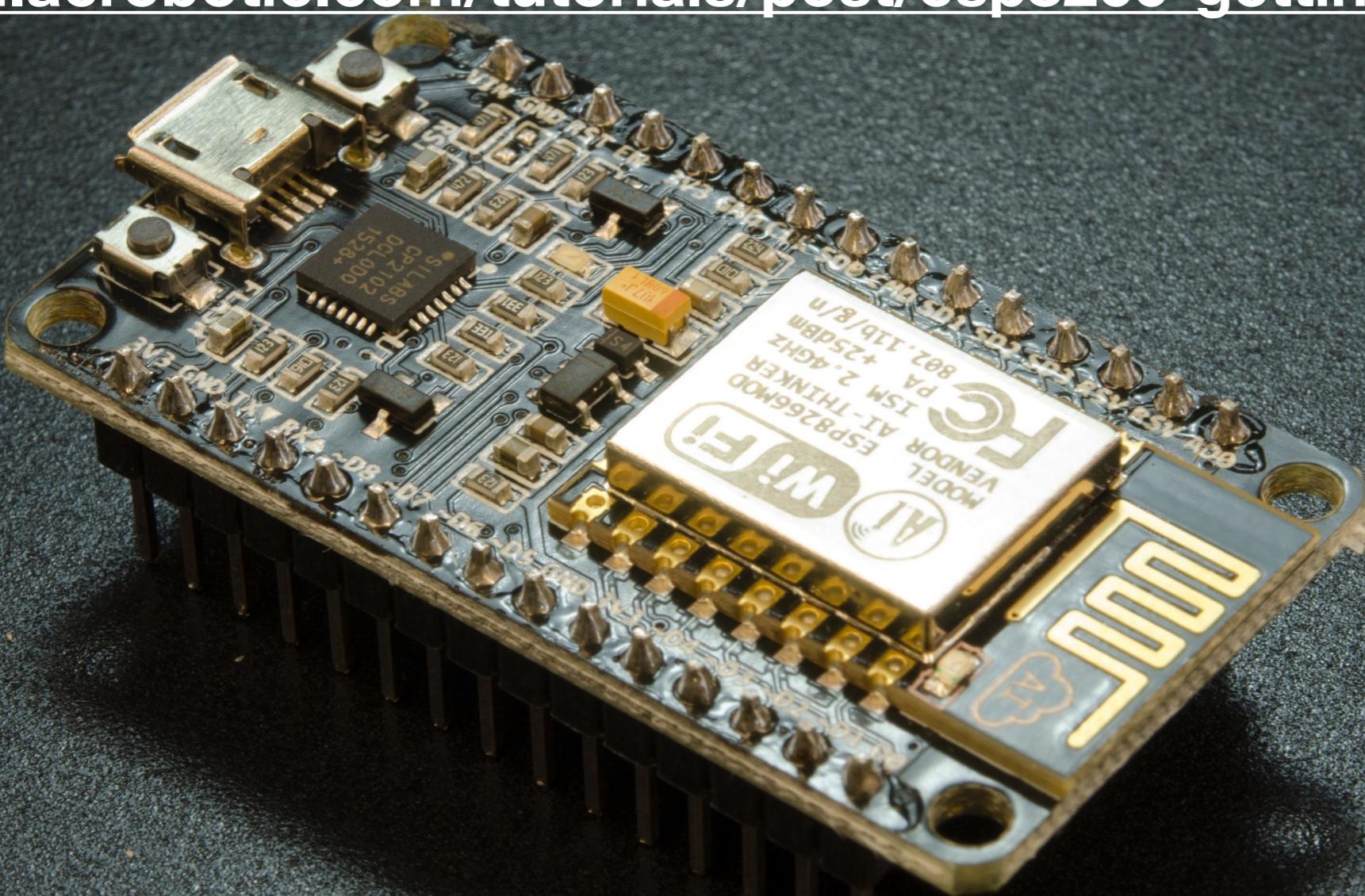
## The ESP8266

Serial to Wi-Fi adapter by Expressive Systems (Summer '14)

ESP-NN 'Breakout' Modules

Development Board for ESP-12E Module More info:

<http://learn.acrobotic.com/tutorials/post/esp8266-getting-started>

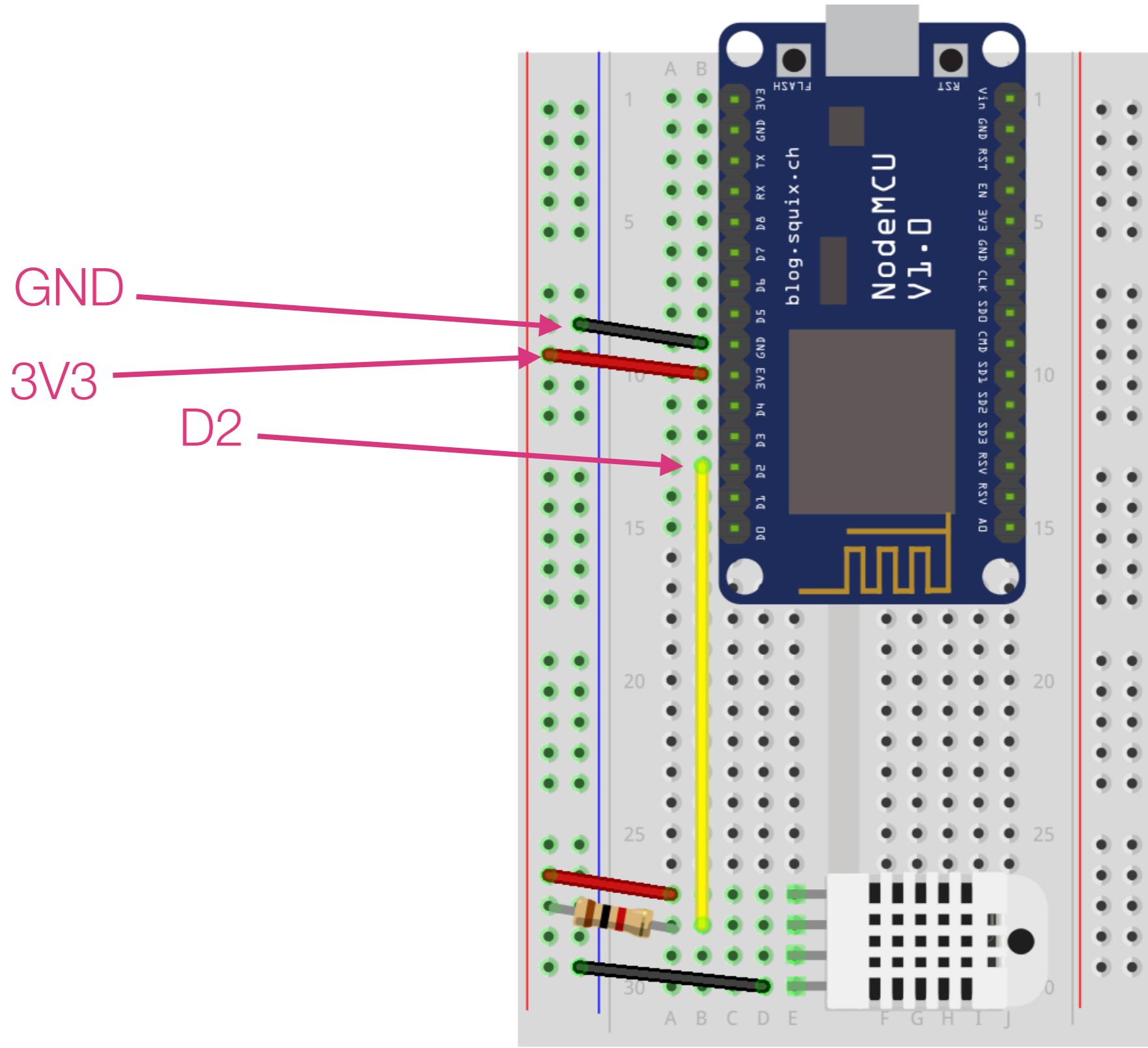


# **Setting Up the Hardware**

Wiring a temp/humidity sensor

# Setting Up the Hardware

## Wiring up the circuit



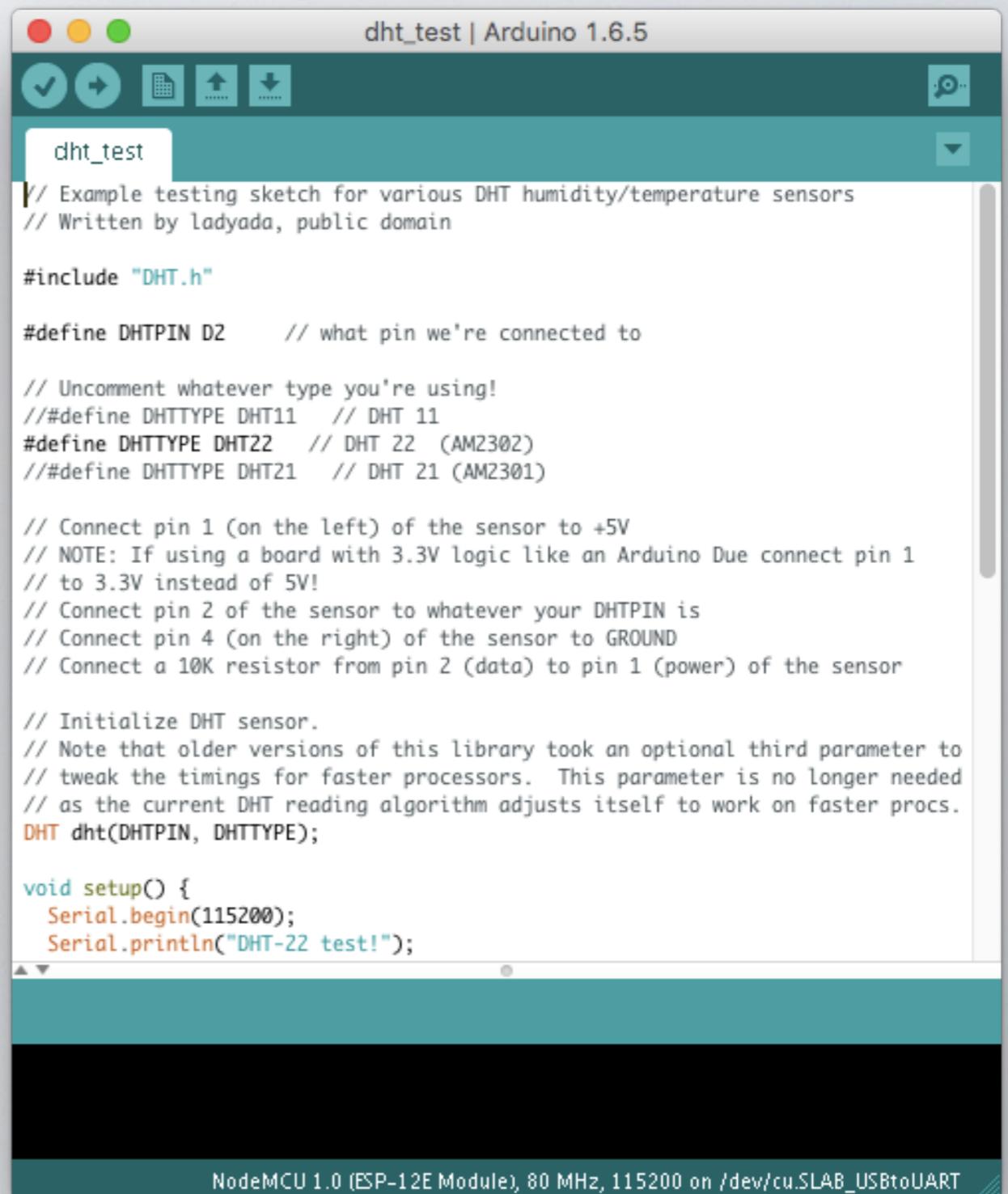
# Getting Started with Arduino

**Installing the Arduino IDE**—a computer application to edit, compile, and upload our programs, as well as communicate via USB with the ESP8266 development board (and others)

[Windows](#)

[Mac \(OSX 10.5+\)](#)

[Linux \(32-bit, 64-bit\)](#)



The screenshot shows the Arduino IDE interface with the title bar "dht\_test | Arduino 1.6.5". The main window displays the code for a DHT sensor test sketch. The code includes comments explaining the setup and usage of the DHT library, defining the DHT pin as D2, and initializing the DHT sensor. It also includes a setup function that begins serial communication at 115200 baud and prints a message to the serial monitor.

```
// Example testing sketch for various DHT humidity/temperature sensors
// Written by ladyada, public domain

#include "DHT.h"

#define DHTPIN D2      // what pin we're connected to

// Uncomment whatever type you're using!
//#define DHTTYPE DHT11    // DHT 11
#define DHTTYPE DHT22    // DHT 22 (AM2302)
//#define DHTTYPE DHT21    // DHT 21 (AM2301)

// Connect pin 1 (on the left) of the sensor to +5V
// NOTE: If using a board with 3.3V logic like an Arduino Due connect pin 1
// to 3.3V instead of 5V!
// Connect pin 2 of the sensor to whatever your DHTPIN is
// Connect pin 4 (on the right) of the sensor to GROUND
// Connect a 10K resistor from pin 2 (data) to pin 1 (power) of the sensor

// Initialize DHT sensor.
// Note that older versions of this library took an optional third parameter to
// tweak the timings for faster processors. This parameter is no longer needed
// as the current DHT reading algorithm adjusts itself to work on faster procs.
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(115200);
  Serial.println("DHT-22 test!");
```

NodeMCU 1.0 (ESP-12E Module), 80 MHz, 115200 on /dev/cu.SLAB\_USBtoUART

# Getting Started with the Arduino IDE

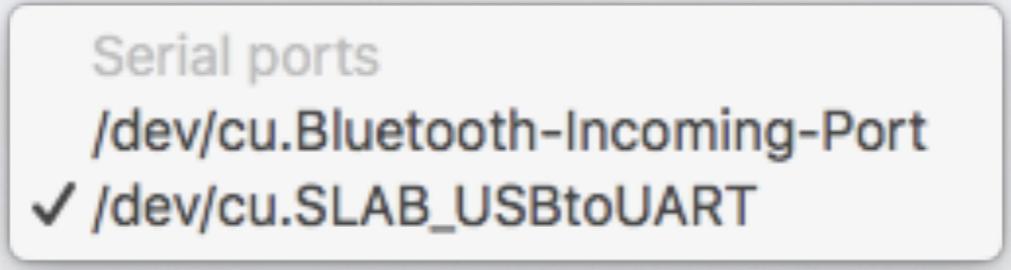
## Configuring the Arduino IDE to support the ESP8266

Download and install the USB drivers:

**<http://j.mp/ESP8266-driver>**

In the Arduino IDE check under:

*Tools > Port*



Serial ports

/dev/cu.Bluetooth-Incoming-Port

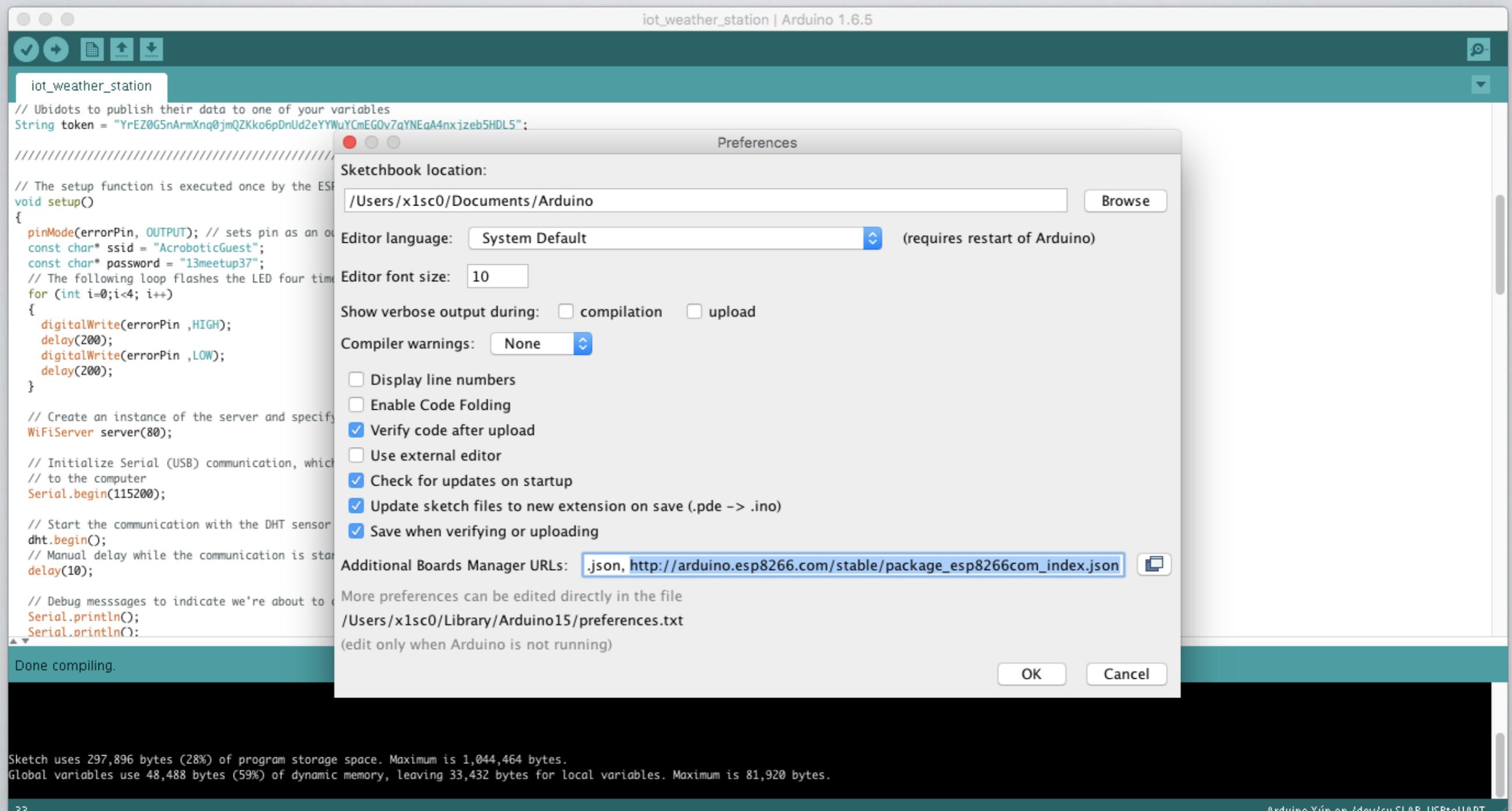
✓ /dev/cu.SLAB\_USBtoUART

# Getting Started with the Arduino IDE

## Configuring the Arduino IDE to support the ESP8266

Navigate to “Preferences” and under Additional Board Manager URLs enter:

**[http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)**



Sketch uses 297,896 bytes (28%) of program storage space. Maximum is 1,044,464 bytes.  
Global variables use 48,488 bytes (59%) of dynamic memory, leaving 33,432 bytes for local variables. Maximum is 81,920 bytes.

# Getting Started with the Arduino IDE

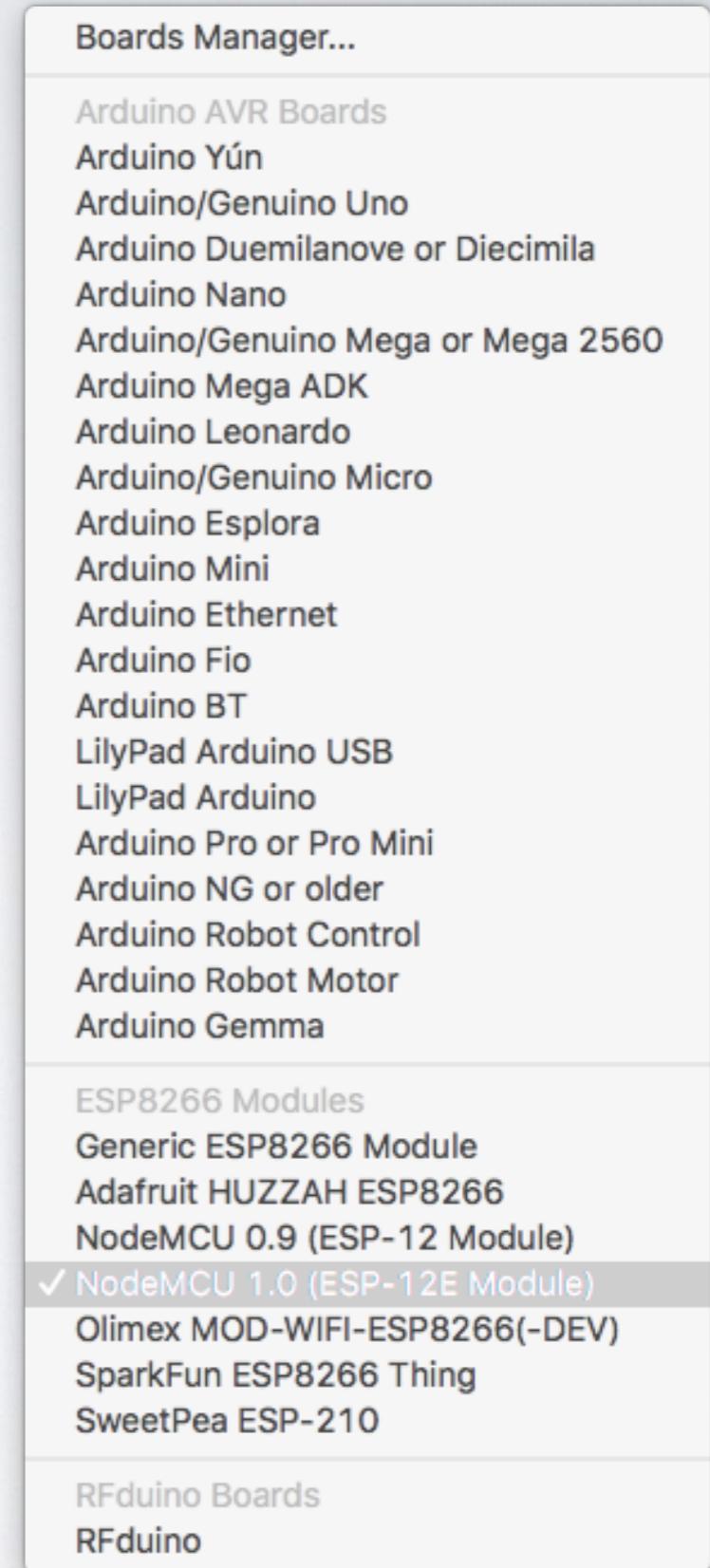
## Configuring the Arduino IDE to support the ESP8266

Install the ESP8266 boards under:

*Tools > Board > Boards Manager*

In the Arduino IDE select:

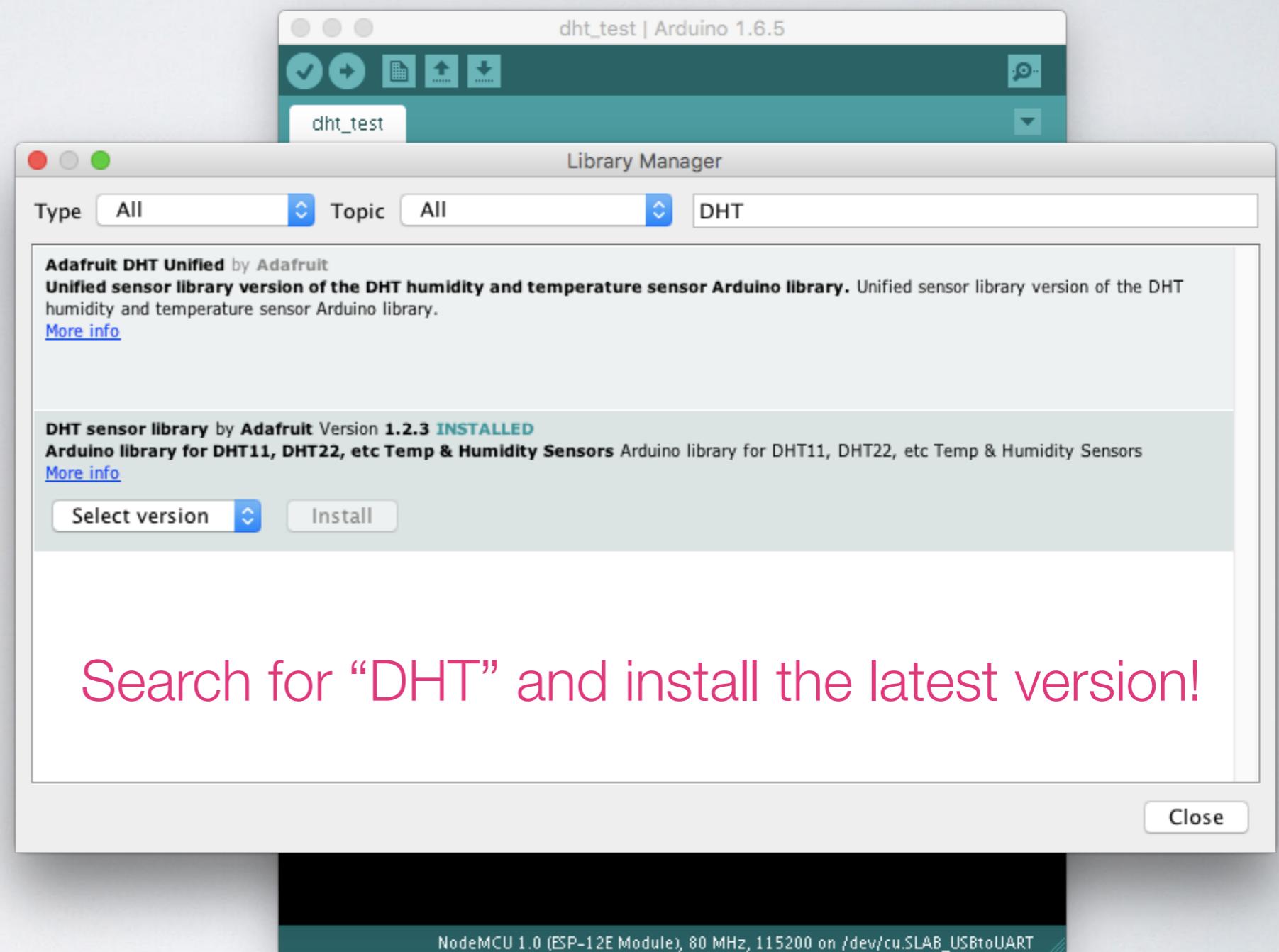
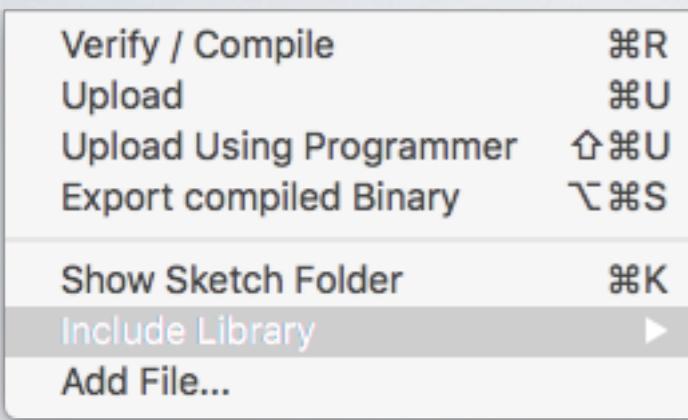
*Tools > Board > NodeMCU 1.0 (ESP-12E)*



# Getting Started with the Arduino IDE

## Configuring the Arduino IDE to support the DHT-22 sensor

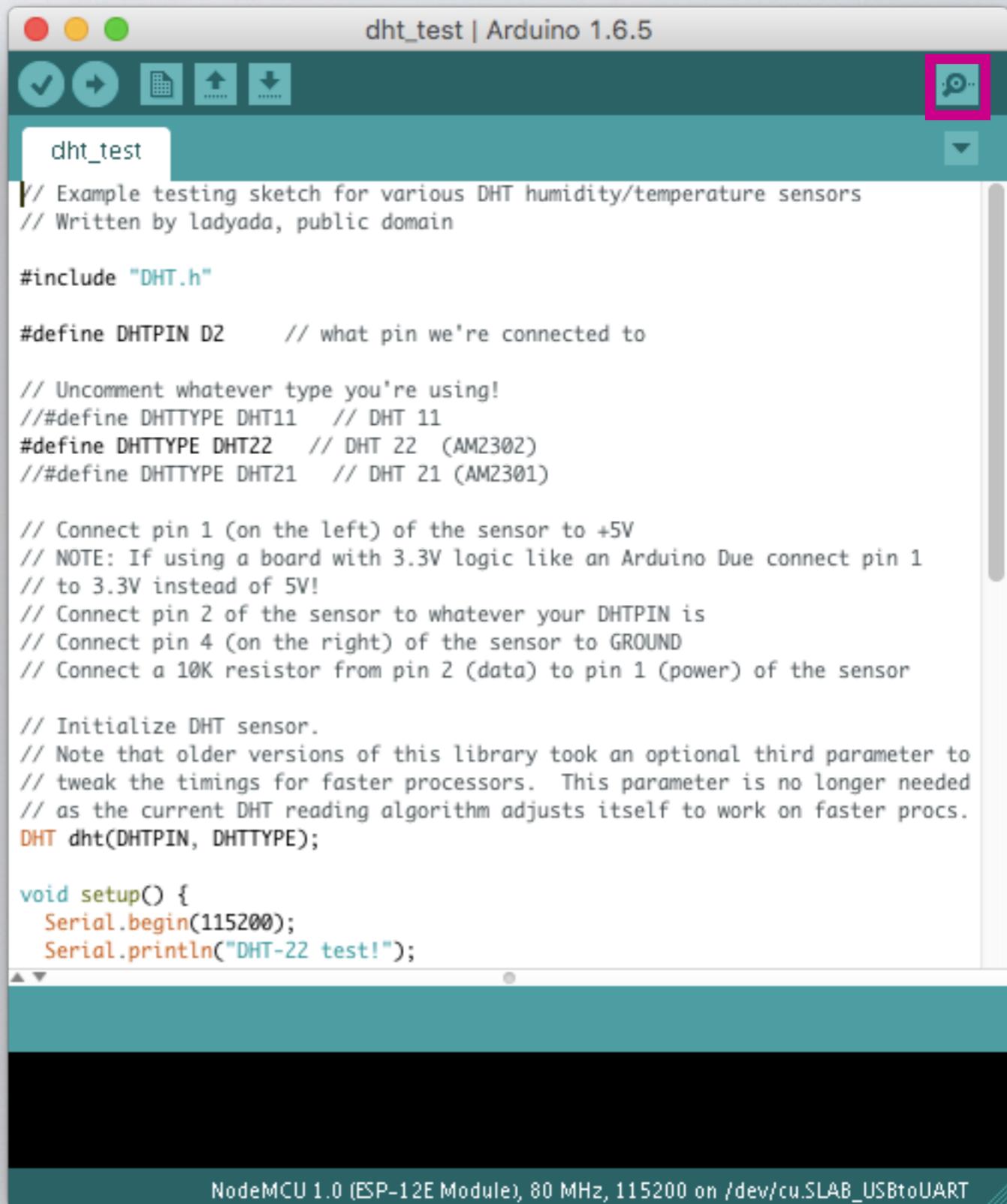
Open the Library Manager from the menu: *Sketch > Include Library > Manage Libraries...*



# Setting Up the Hardware

## Testing our connections [iot\_weather\_station/dht\_test/dht\_test.ino]

Upload dht\_test.ino to the board  
and open the Serial Monitor to  
check the measured data!



The screenshot shows the Arduino IDE interface with the sketch 'dht\_test' loaded. The code is a DHT sensor testing sketch. It includes comments for connecting the sensor to pins D2, D11, D22, or D21. It initializes the DHT sensor and sets up the serial port at 115200 baud to print DHT-22 test data. The code is as follows:

```
// Example testing sketch for various DHT humidity/temperature sensors
// Written by ladyada, public domain

#include "DHT.h"

#define DHTPIN D2      // what pin we're connected to

// Uncomment whatever type you're using!
//#define DHTTYPE DHT11 // DHT 11
#define DHTTYPE DHT22 // DHT 22 (AM2302)
//#define DHTTYPE DHT21 // DHT 21 (AM2301)

// Connect pin 1 (on the left) of the sensor to +5V
// NOTE: If using a board with 3.3V logic like an Arduino Due connect pin 1
// to 3.3V instead of 5V!
// Connect pin 2 of the sensor to whatever your DHTPIN is
// Connect pin 4 (on the right) of the sensor to GROUND
// Connect a 10K resistor from pin 2 (data) to pin 1 (power) of the sensor

// Initialize DHT sensor.
// Note that older versions of this library took an optional third parameter to
// tweak the timings for faster processors. This parameter is no longer needed
// as the current DHT reading algorithm adjusts itself to work on faster procs.
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(115200);
  Serial.println("DHT-22 test!");
```

NodeMCU 1.0 (ESP-12E Module), 80 MHz, 115200 on /dev/cu.SLAB\_USBtoUART

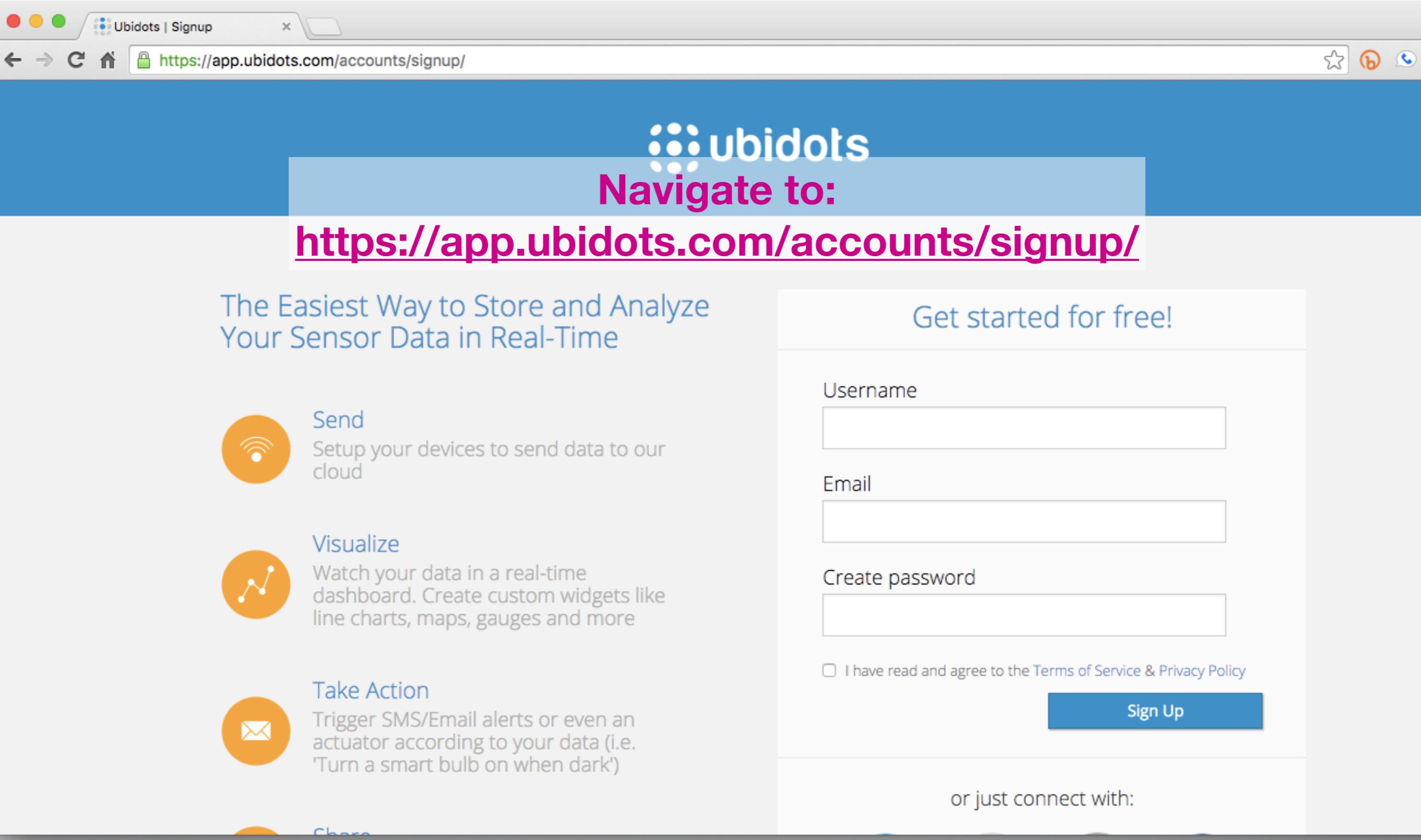
# **Configuring the Backend and Frontend**

Using Ubidots API for posting data (backend)

Using Ubidots Widgets for displaying data (frontend)

# Configuring the Backend

## Getting started with Ubidots



The screenshot shows a web browser window for the Ubidots Signup page (<https://app.ubidots.com/accounts/signup/>). The page features a blue header with the Ubidots logo and navigation icons. Below the header, a large call-to-action button with a white background and pink text reads "Navigate to: <https://app.ubidots.com/accounts/signup/>". To the left of this button, there's a section titled "The Easiest Way to Store and Analyze Your Sensor Data in Real-Time" with three orange circular icons: "Send" (Wi-Fi), "Visualize" (Line chart), and "Take Action" (Email). On the right side, there's a "Get started for free!" form with fields for "Username", "Email", and "Create password", each accompanied by a corresponding input field. At the bottom of the form is a checkbox for accepting terms and conditions, followed by a "Sign Up" button. Below the form, there's a link for social media connection.

Ubidots | Signup

<https://app.ubidots.com/accounts/signup/>

ubidots

Navigate to:

<https://app.ubidots.com/accounts/signup/>

The Easiest Way to Store and Analyze Your Sensor Data in Real-Time

Send

Setup your devices to send data to our cloud

Visualize

Watch your data in a real-time dashboard. Create custom widgets like line charts, maps, gauges and more

Take Action

Trigger SMS/Email alerts or even an actuator according to your data (i.e. 'Turn a smart bulb on when dark')

Get started for free!

Username

Email

Create password

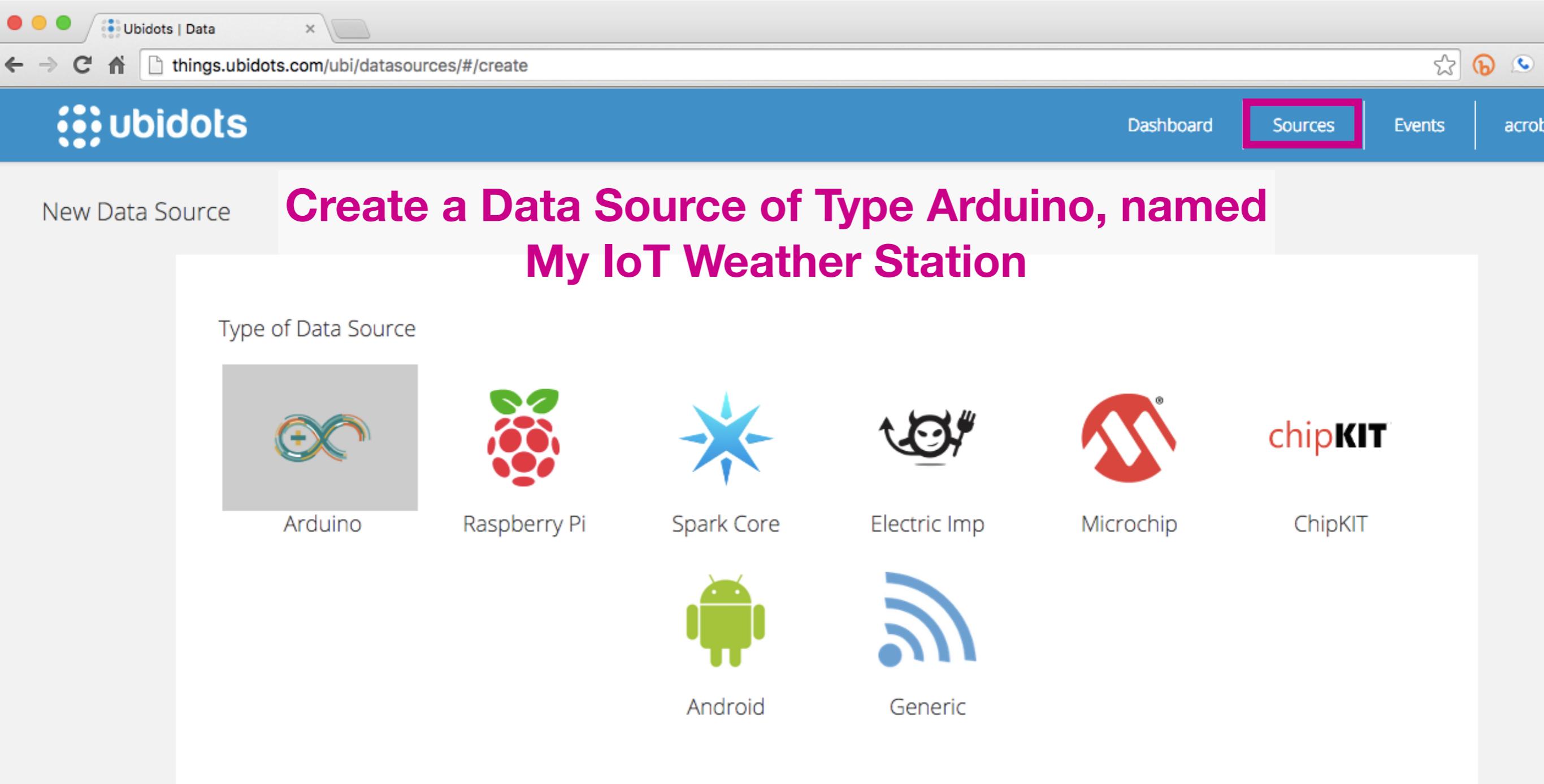
I have read and agree to the [Terms of Service & Privacy Policy](#)

Sign Up

or just connect with:

# Configuring the Backend

## Setting up a Data Source



The screenshot shows a web browser window for the Ubidots platform at [things.ubidots.com/ubi/datasources/#/create](https://things.ubidots.com/ubi/datasources/#/create). The browser's address bar and various icons are visible at the top. The main content area has a blue header with the Ubidots logo and navigation links for Dashboard, Sources (which is highlighted with a pink border), Events, and acrob. A sub-header on the left says "New Data Source". The central part of the page displays the title "Create a Data Source of Type Arduino, named My IoT Weather Station" in large, bold, magenta text. Below this, there is a section titled "Type of Data Source" with a grid of icons and labels for different hardware types:

- Arduino (represented by a green and orange gear icon)
- Raspberry Pi (represented by a red Raspberry Pi logo icon)
- Spark Core (represented by a blue starburst icon)
- Electric Imp (represented by a black devil-like face icon)
- Microchip (represented by a red Microchip logo icon)
- ChipKIT (represented by a red ChipKIT logo icon)
- Android (represented by a green Android robot icon)
- Generic (represented by a blue Wi-Fi signal icon)

# Configuring the Backend

## Setting up variables for the Data Source

The screenshot shows the Ubidots web interface for managing data sources. The top navigation bar includes links for Dashboard, Sources, Events, and acrobatics. The main content area is titled "Create two ‘normal’ variables: Temperature (F) and Humidity (%)".

**Left Sidebar:**

- Name:** My ESP8266
- Variables:** 0 Variables (No Last Activity)
- Tags:** No tags
- Description:** (empty)
- Context:** \_icon: arduino
- URL Endpoint:** (empty)

**Main Area:**

- Add Variable** button
- f(x)** placeholder
- Add Derived Variable** button

**Right Sidebar:**

- What are Variables?**

A variable is a series of data containing values that change over time. In our case, variables refer to sensors that measure something from the real world, such as:

  - Temperature
  - Humidity
  - Pressure
  - Position
  - Speed
  - Acceleration
  - Sound Level
  - Light Level
  - Amount of people
  - Volume
  - Water Level
  - Motion Presence
  - Disk Usage
  - .. or any time-stamped data!
- Ubidots accepts any type of time

# Configuring the Backend

## Getting your Ubidots access token

The screenshot shows a web browser window for 'acrobotic's profile' at 'things.ubidots.com/userdata/api/'. The main content is the 'API Access' section under 'Settings'. It displays an 'API Key' table with one row containing the value '4b87eeabda9815fe4cb013651ec4382a53608dcb'. Below this is the 'Authentications Tokens' section, which includes a 'Create Short Token' button and a 'Create Token' button. A new token 'newToken' is listed with the value 'YrEZ0G5nArmXnq0jmQZKko6pDnUd2eYYWuYCmEGOv7qYNEqA4nxjzeb5HDL5'. The 'API Keys' tab in the sidebar is highlighted.

acrobotic's profile. things.ubidots.com/userdata/api/ acrob

ubidots

Dashboard Sources Events acrob

Settings

My account

API Keys

Plans and Billing

Account Usage

API Access

This is your account's unique and fixed master key.

It's only purpose is to request tokens through the [Auth API endpoint](#), which are then used in every API request.

API Key	Value
	4b87eeabda9815fe4cb013651ec4382a53608dcb

Authentications Tokens

Alternatively, you can create and revoke auth tokens from this interface:

Create Short Token Create Token

newToken YrEZ0G5nArmXnq0jmQZKko6pDnUd2eYYWuYCmEGOv7qYNEqA4nxjzeb5HDL5

# Configuring the ESP8266

## Change the Wi-Fi settings, Token, and Variable IDs

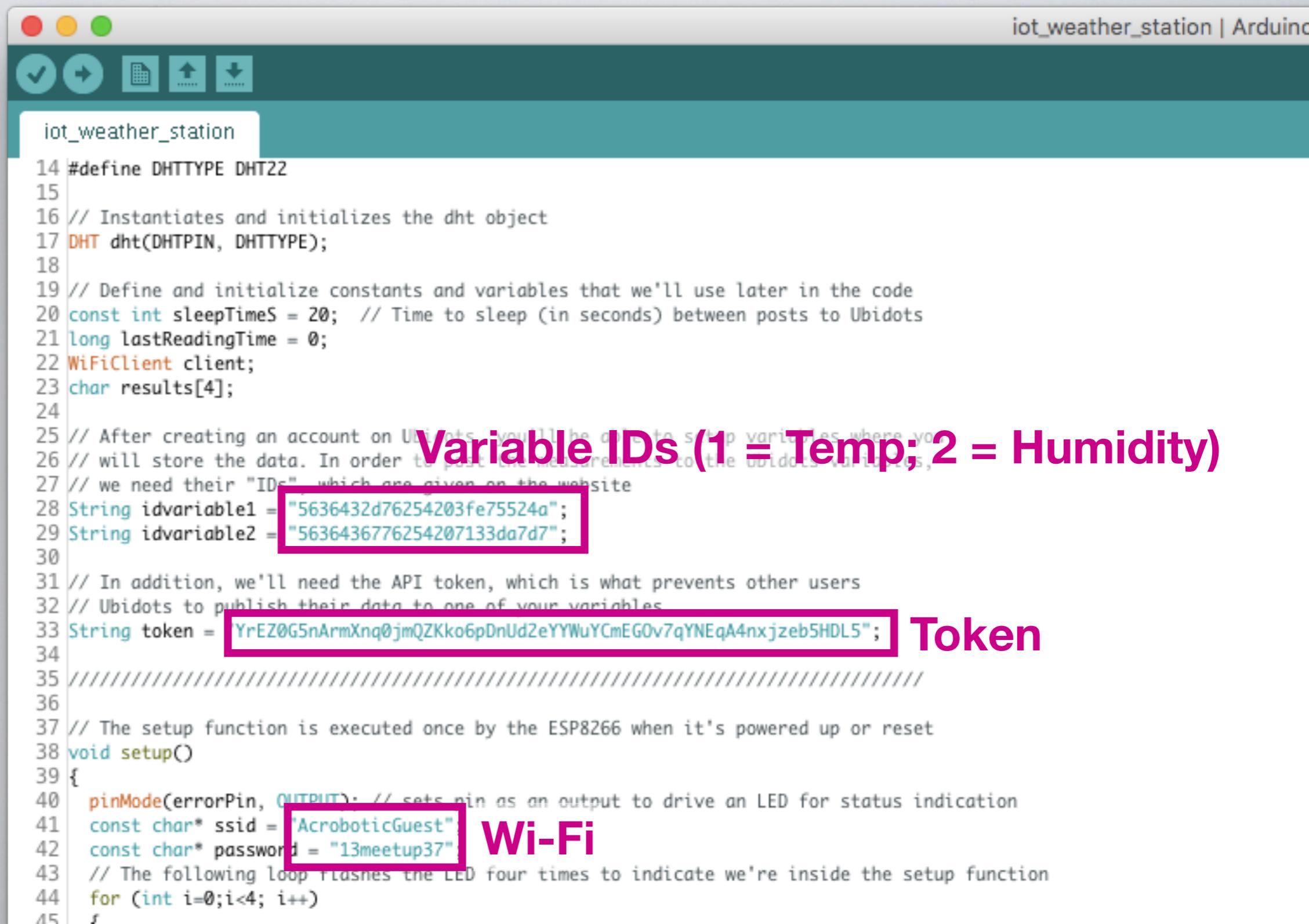
The screenshot shows the Ubidots Data interface. At the top, there's a browser-like header with tabs for 'Ubidots | Data' and a URL 'things.ubidots.com/ubi/datasources/#/detail/56364747762542284d39dd56/variable/563647fe7625422c832b92de'. Below the header, the Ubidots logo is on the left, and navigation links for 'Dashboard', 'Sources', 'Events', and 'acrob...' are on the right.

In the main content area, there's a message 'This variable is empty' with a sub-instruction 'Check out our API docs to learn how to send data to your variables.' To the left, there's a sidebar with a yellow button labeled 'Temperature' and a 'Temperature' section containing fields for 'Name' (set to 'Temperature'), 'Unit' (set to 'F'), 'Description' (empty), 'Tags' (empty), and 'ID' (empty).

A tooltip is visible over the 'Variable ID' field in the sidebar, displaying the value '563647fe7625422c832b92de'. The overall theme is light blue and white, with orange highlights for interactive elements like the upload button and sidebar sections.

# Configuring the ESP8266

## Change the Wi-Fi settings, Token, and Variable IDs



The screenshot shows the Arduino IDE interface with the sketch titled "iot\_weather\_station". The code is written in C++ and defines constants for DHT type, sleep time, and Ubidots variable IDs, along with a Ubidots API token and Wi-Fi credentials.

```
14 #define DHTTYPE DHT22
15
16 // Instantiates and initializes the dht object
17 DHT dht(DHTPIN, DHTTYPE);
18
19 // Define and initialize constants and variables that we'll use later in the code
20 const int sleepTimeS = 20; // Time to sleep (in seconds) between posts to Ubidots
21 long lastReadingTime = 0;
22 WiFiClient client;
23 char results[4];
24
25 // After creating an account on Ubidots, you'll be able to setup variables where you
26 // will store the data. In order to post the measurements to the ubidots variables,
27 // we need their "IDs", which are given on the website
28 String idvariable1 = "5636432d76254203fe75524a";
29 String idvariable2 = "5636436776254207133da7d7";
30
31 // In addition, we'll need the API token, which is what prevents other users
32 // Ubidots to publish their data to one of your variables
33 String token = "YrEZ0G5nArmXnq0jmQZKko6pDnUd2eYYWuYCmEG0v7qYNEqA4nxjzeb5HDL5";
34
35 ///////////////////////////////////////////////////
36
37 // The setup function is executed once by the ESP8266 when it's powered up or reset
38 void setup()
39 {
40     pinMode(errorPin, OUTPUT); // sets pin as an output to drive an LED for status indication
41     const char* ssid = "AcroboticGuest";
42     const char* password = "13meetup37";
43     // The following loop flashes the LED four times to indicate we're inside the setup function
44     for (int i=0;i<4; i++)
45     {
46         if (i>0)
47             delay(500);
48         digitalWrite(errorPin, HIGH);
49         delay(50);
50         digitalWrite(errorPin, LOW);
51         delay(50);
52     }
53 }
```

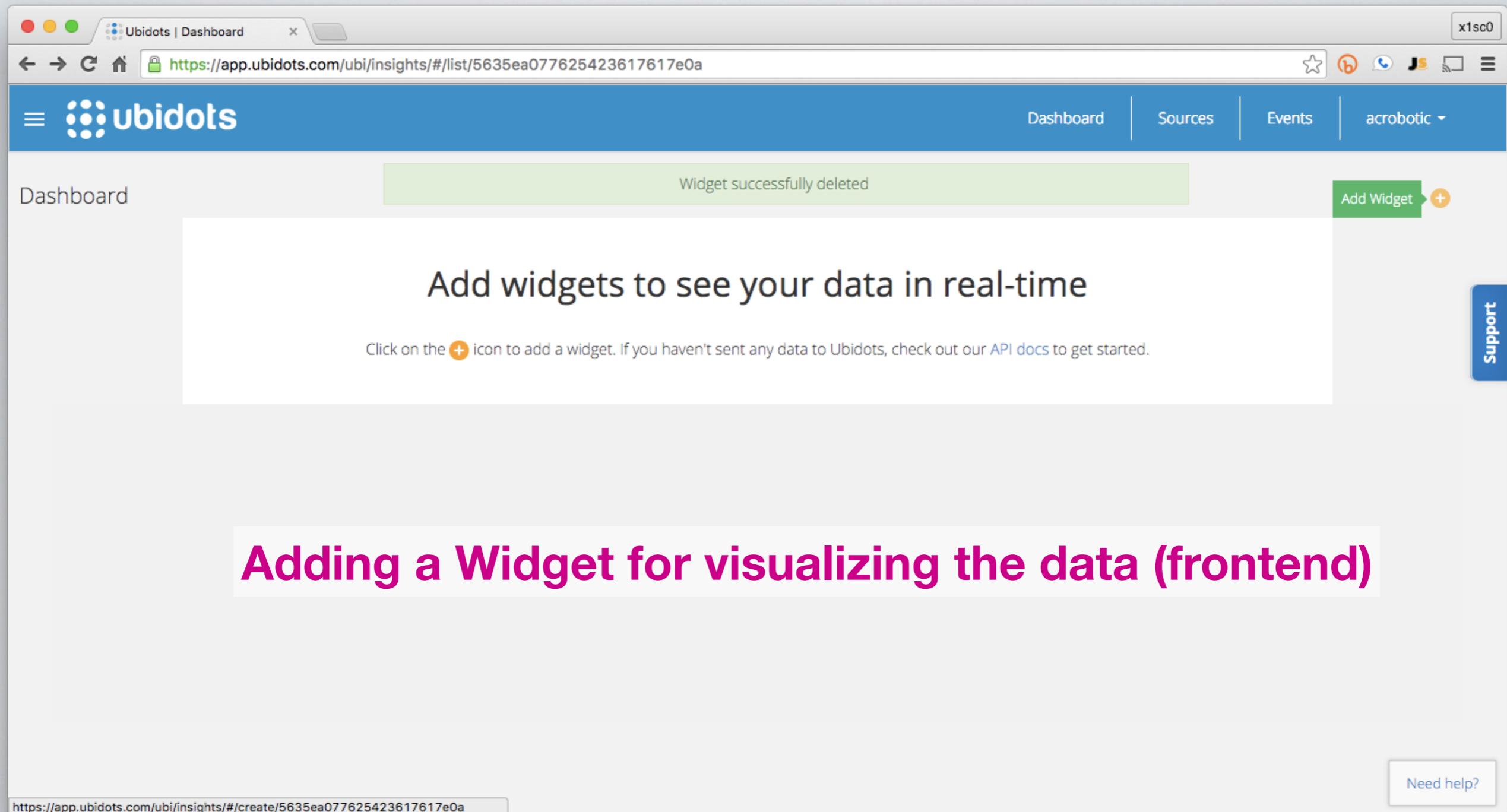
**Variable IDs (1 = Temp; 2 = Humidity)**

**Token**

**Wi-Fi**

# Visualizing the Data

## Configuring the Ubidots fronted widgets for data visualization



The screenshot shows a web browser window for the Ubidots Dashboard at the URL <https://app.ubidots.com/ubi/insights/#/list/5635ea077625423617617e0a>. The dashboard has tabs for 'Dashboard', 'Sources', 'Events', and a dropdown for 'acrobatic'. A green success message box says 'Widget successfully deleted'. Below it, a large white area says 'Add widgets to see your data in real-time' with a note: 'Click on the + icon to add a widget. If you haven't sent any data to Ubidots, check out our [API docs](#) to get started.' A blue 'Support' button is on the right. A pink callout box highlights the text 'Adding a Widget for visualizing the data (frontend)'. The browser status bar at the bottom shows the same URL.

Widget successfully deleted

Add widgets to see your data in real-time

Click on the + icon to add a widget. If you haven't sent any data to Ubidots, check out our [API docs](#) to get started.

Support

Need help?

Adding a Widget for visualizing the data (frontend)

# Visualizing the Data

## Configuring the Ubidots fronted widgets for data visualization

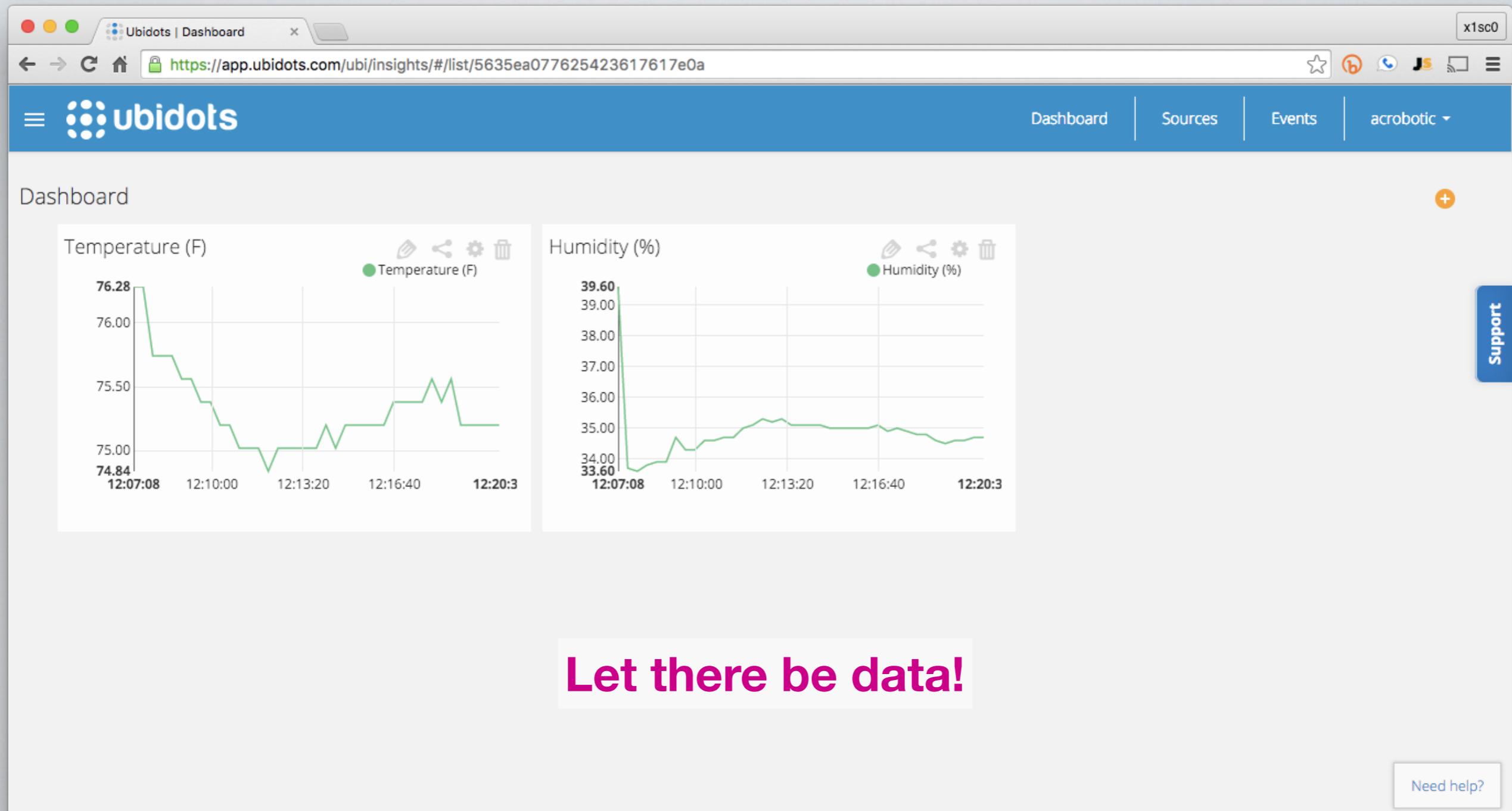
The screenshot shows the Ubidots Dashboard interface. At the top, there's a browser header with the URL <https://app.ubidots.com/ubi/insights/#/create/5635ea077625423617617e0a>. Below the header, the Ubidots logo is on the left, and a navigation bar with tabs for Dashboard, Sources, Events, and a dropdown menu for 'acrobotic'. A 'Support' button is located on the right side of the header.

The main content area is titled 'New Widget' and features a large, bold heading 'Using Line Charts for time-series data (Temp/Hum vs. Time)' in pink. Below this, a sub-instruction reads 'How do you want to visualize your widget?'. A grid of twelve icons represents different visualization types:

- Statement
- Line Chart
- Metric
- Scatter Plot
- Map
- GPS Trace
- Multi-markers Map
- Multi-line Chart (highlighted with a green callout bubble: 'Add a simple line chart showing the last data-points of a variable')
- Gauge
- Indicator
- Switch
- Table Values
- Table Historical

# Visualizing the Data

## Configuring the Ubidots fronted widgets for data visualization



# Visualizing the Data

## Accessing the Ubidots fronted widgets for data visualization

