

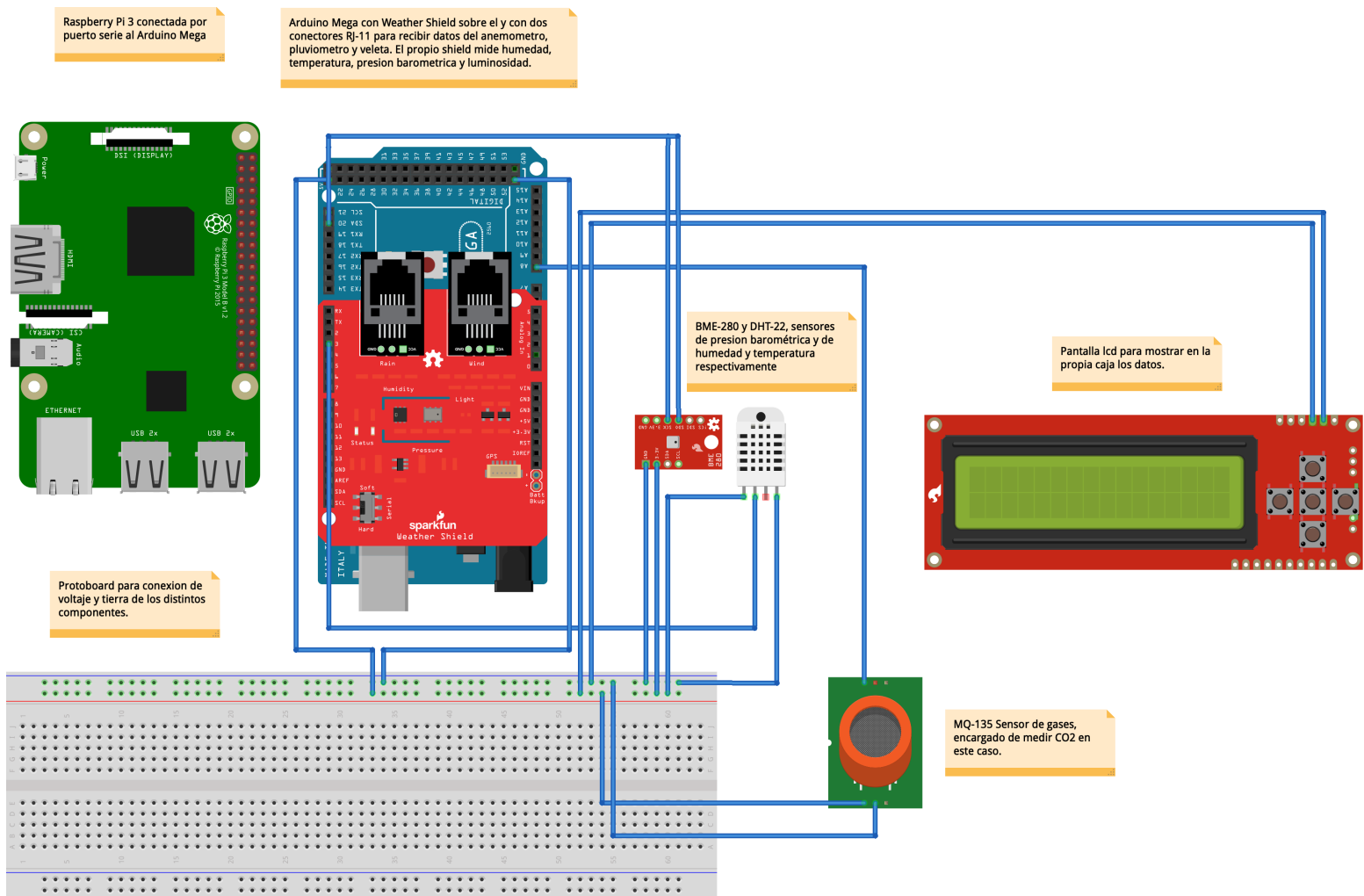
# Desarrollo e implementación de una estación meteorológica

## La guía sencilla

Con esta guía y sin conocimientos previos de electrónica ni programación serás capaz de construir una estación meteorológica totalmente funcional, y de paso aprenderás muchas cosas nuevas, vamos allá.

En primer lugar la lista con el material necesario, puedes variarlo en función a tus necesidades y/o conocimientos pero para seguir la guía sin experiencia previa sería recomendable utilizar los mismos componentes.

- Arduino Mega 2560
- Raspberry Pi 3 model B v1.2 y MicroSD de 16 Gb para el S.O
- Sparkfun Weather Shield
- Sparkfun Weather Meters
- BME-280
- DHT-22
- MQ-135
- Pantalla LCD
- Protoboard



fritzing

En el esquema se pueden apreciar los componentes anteriormente mencionados y su conexión. En mi caso usé una LCD 24x4 que no esta pensada para Arduino ya que la teníamos en el almacén, se puede seguir el tutorial en este caso pero lo mas recomendable seria usar un LCD para Arduino con I2C, para evitar cableados engorrosos (explico mas adelante las diferencias entre pantallas con mas detalle).

También al final de las explicaciones dejare algunos links útiles a conceptos que puedan resultar más complicados.

## **Explicación de cada componente:**

Junto a cada componente se encuentra un enlace donde se puede adquirir.

### **- Arduino Mega**

La placa Arduino con mayor conectividad que podemos adquirir, debido a que el Weather Shield va a ocupar gran parte de nuestras conexiones un Arduino Uno no podrá con todo el cableado que vamos a usar.

<https://store.arduino.cc/mega-2560-r3>

### **- Raspberry Pi 3**

Conectado al Arduino mediante el puerto serie usando el cable, proporcionará alimentación al Arduino y recibirá datos a través de dicho puerto serie, los procesara y guardara en una base de datos.

<https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>

### **- Weather Shield**

Shield para Arduino Uno o Mega que nos proporciona temperatura, humedad, presión barométrica y cantidad de luz.

<https://www.sparkfun.com/products/13956>

### **- Weather Meters**

Kit compuesto de un anemómetro, pluviómetro y veleta, además de un mástil para acoplarlos y tornillería varia. Viene con 2 cables Rj-11 a través de los cuales envía los datos que recoge.

<https://www.sparkfun.com/products/8942>

### **- BME-280**

Sensor de presión barométrica, en mi caso uso la versión con 4 conexiones, tierra, voltaje, SDA y SCL.

<https://bit.ly/2ZhjGWP> => AliExpress, link acortado.

### **- DHT-22**

Sensor de temperatura y humedad, lo usaremos para medir la temperatura y humedad ambiente y los integrados en la weather shield mediaran la temperatura y humedad de los componentes.

En mi caso use el que ya viene con los cables integrados.

<https://www.adafruit.com/product/393>

### **- MQ-135**

Sensor de gases, usado en este caso para medir dióxido de carbono, también mide amoníaco, óxidos de nitrógeno, alcohol y benceno. En este caso estará calibrado para CO2 y usara una librería particular para ello.

<https://satkit.com/mq135-mq-135-sensor-calidad-aire-gas-detector-polucion-conta>

### **- Pantalla LCD**

En mi caso estoy utilizando una LCD2004 20x4, usada para impresoras 3d Anet y con un cable de extension MKS, no es específico de Arduino y hay que puentear los pines del cable, lo cual es poco práctico.

En este caso si recomendaría utilizar un LCD distinto al mío, hay muchos disponibles para Arduino, dejo aquí un ejemplo:

### **Tutorial de montaje y uso:**

En él se explica con detalle cómo usar el LCD y los componentes requeridos.

<https://www.prometec.net/bus-i2c/>

### **- Protoboard**

También llamada placa de pruebas, es un tablero con orificios conectados entre sí internamente en línea, usado para expandir la tierra y el voltaje desde Arduino.

Necesitaremos además cables puente para usar la protoboard, lo más recomendable es adquirir todo en un pack de iniciación, dejo este para Arduino Mega:

<https://amzn.to/30vmZH6>

## **Enlaces útiles para comprender mejor lo que estamos haciendo:**

### **- ¿ Qué es el bus i2c ?:**

<https://www.luisllamas.es/arduino-i2c/>

### **- ¿ Qué es una shield para Arduino ?**

<https://aprendiendoarduino.wordpress.com/2015/03/23/shields-para-arduino/>

- ¿ Qué quiero decir cuando me refiero a voltaje, tierra, conexiones en serie, resistencias, condensadores... ?

|

|

——>

### **Tutorial básico de electrónica**

[https://profesormolina1.webcindario.com/tutoriales/componentes/tut\\_comp.htm](https://profesormolina1.webcindario.com/tutoriales/componentes/tut_comp.htm)

### **Alimentando tu Arduino**

<https://www.geekfactory.mx/tutoriales/tutoriales-arduino/alimentar-el-arduino-la-guia-definitiva/>

### **Algo mas de electrónica nunca viene mal**

Primer capítulo, aunque todos son recomendables.

<https://www.youtube.com/watch?v=LjYClvMPRdE>

# Paso 1 - Comenzamos

Una vez obtenidos los materiales y si no se disponía de conocimiento de conceptos de electrónica leídos los tutoriales (es altamente recomendable para entender lo que vamos a hacer) empezamos a construir nuestra estación.

El software necesario será:

- Arduino IDE
- Distribución Linux, en mi caso Ubuntu 18.04.2 LTS
- IDE o editor de texto de tu preferencia (Visual Studio en mi caso)

Obviamente nuestro primer paso es instalar Ubuntu si no lo tenemos instalado, la propia página de Ubuntu ofrece un gran tutorial paso a paso:

Primero almacenar el S.O en un pendrive para poder arrancar el PC desde él.

## **Tutorial windows:**

<https://tutorials.ubuntu.com/tutorial/tutorial-create-a-usb-stick-on-windows#0>

## **Tutorial MacOS:**

<https://tutorials.ubuntu.com/tutorial/tutorial-create-a-usb-stick-on-macos#0>

Una vez terminada la guía anterior seguimos esta y tendremos instalado nuestra distribución de Linux.

<https://tutorials.ubuntu.com/tutorial/tutorial-install-ubuntu-desktop#0>

Si todo ha ido bien, ya tendremos nuestro Ubuntu instalado y podremos empezar a descargar las herramientas necesarias para trabajar.

El primer paso para desarrollar en Arduino es descargar su IDE (entorno de desarrollo integrado, se usa para programar, compilar y subir el código al propio Arduino, lo explicaremos mas en detalle mas adelante) de la propia página:

<https://www.arduino.cc/en/Main/Software>



Ahora usando la terminal (Control + alt + T en Ubuntu) nos movemos al directorio descargas (dependerá del idioma del S.O) usando el comando `cd`, **c**hange **d**irectory.

```
cd ~/Downloads
```

Descomprimos el fichero `.tar.xz` con el comando `tar -xvf`.

```
tar -xvf arduino-1.6.6-*.tar.xz
```

Movemos con permiso de superusuario a la carpeta **/opt/** para poder usarlo desde cualquier parte.

```
sudo mv arduino-1.6.6 /opt
```

Cambiamos de directorio a donde lo hemos movido previamente.

```
cd /opt/arduino-1.6.6/
```

Le damos permiso de ejecución para poder instalarlo. Más información sobre el comando **chmod** :

<https://desarrolloweb.com/articulos/tutorial-comando-chmod.html>

```
chmod +x install.sh
```

Y lo instalamos con ejecutándolo.

```
./install.sh
```

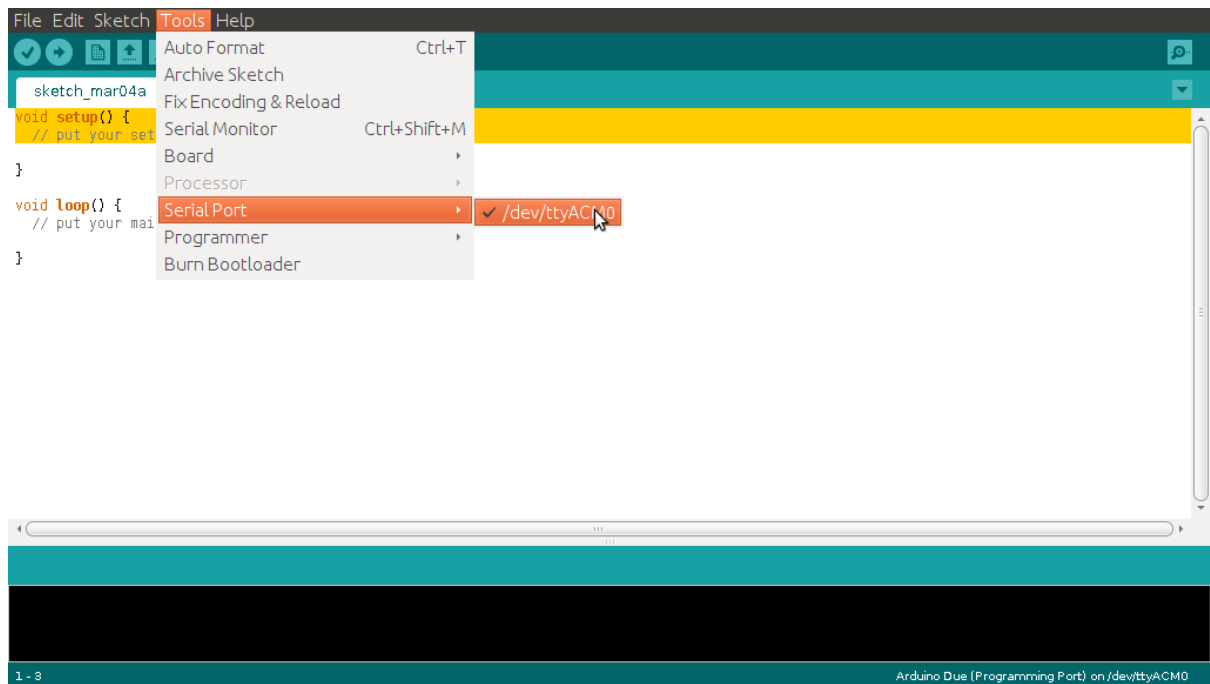
Ya podemos lanzar el IDE desde el icono creado en el escritorio, la primera vez nos dirá que el autor es desconocido, pero aceptamos y podremos usarlo a partir de ahora sin problemas.

Una vez instalado podemos comprobar si nuestro Arduino funciona correctamente subiéndole un proyecto de ejemplo llamado Blink, el cual hará parpadear un led, si bien es simple con esta prueba comprobamos el cable al puerto serie, los puertos del PC y la propia placa.

Antes de poder subir el programa a Arduino debemos habilitar los puertos en Ubuntu, ejecutando el comando:

```
$ sudo usermod -a -G dialout <username>  
$ sudo chmod a+rw /dev/ttyUSB0
```

Dónde <username> es nuestro nombre de usuario en Ubuntu y / **dev/ttyUSB0** es el nombre del puerto por el que nos conectamos, para saber que puerto estamos usando nos vamos al Arduino IDE, herramientas, puerto serie y ahí podemos comprobar el nombre.



Ahora sí podemos seguir uno de los siguientes tutoriales para comprobar que todo esta correcto.

<https://www.arduino.cc/en/tutorial/blink> => Tutorial oficial en inglés.

<https://aprendiendoarduino.wordpress.com/2016/06/26/primer-proyecto-blink-2/> => En español

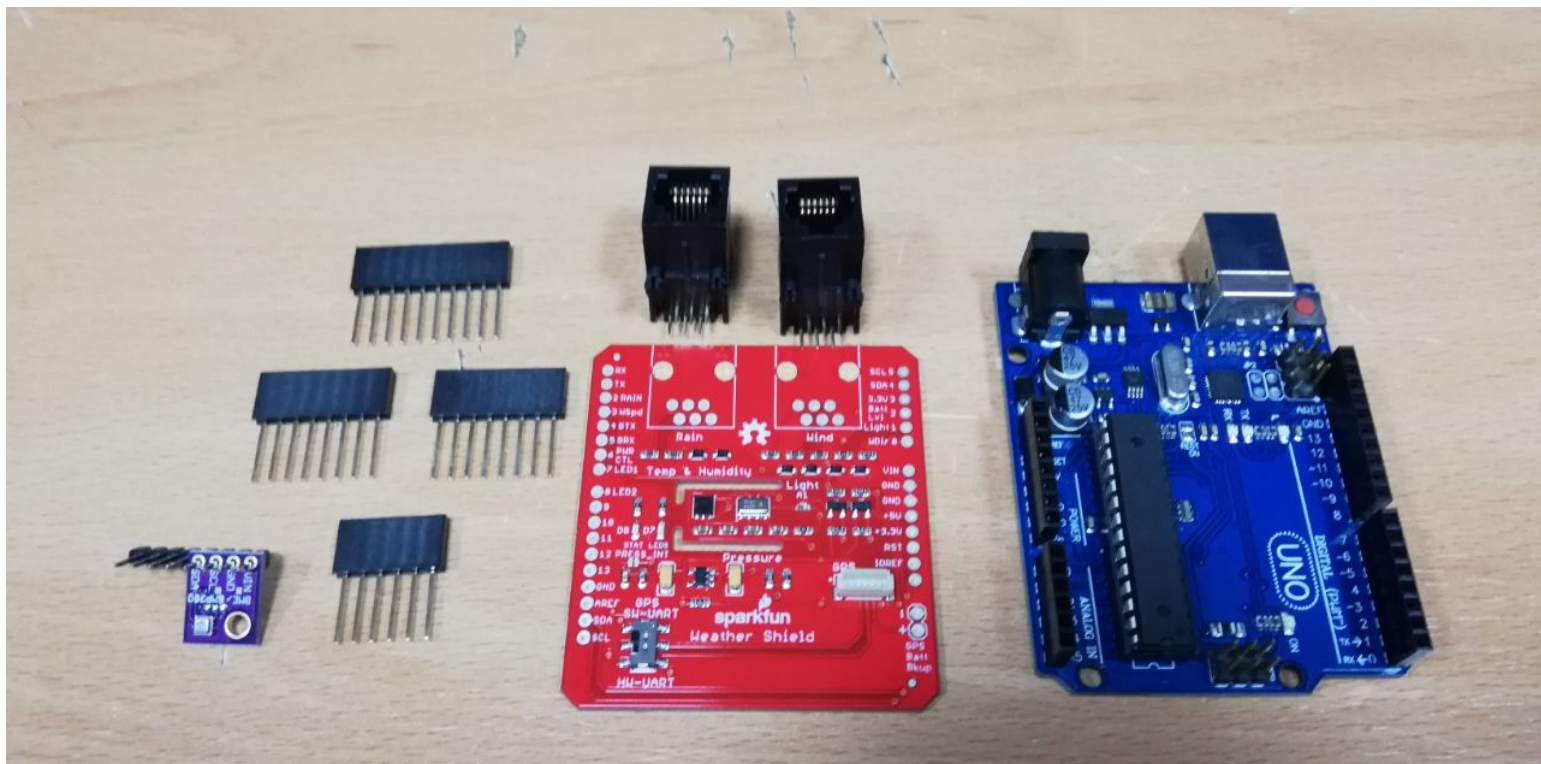
Una vez que todo esta configurado y listo, pasamos a ensamblar nuestra estación.

## Paso 2 - Ensamblado

Material necesario:

- Soldador con base
- Estaño
- Pines de expansion (Stacking headers)

En primer lugar vamos a acoplar el Weather Shield a nuestro Arduino, las fotos son de un Arduino Uno porque en un principio no necesitábamos tanta conectividad.



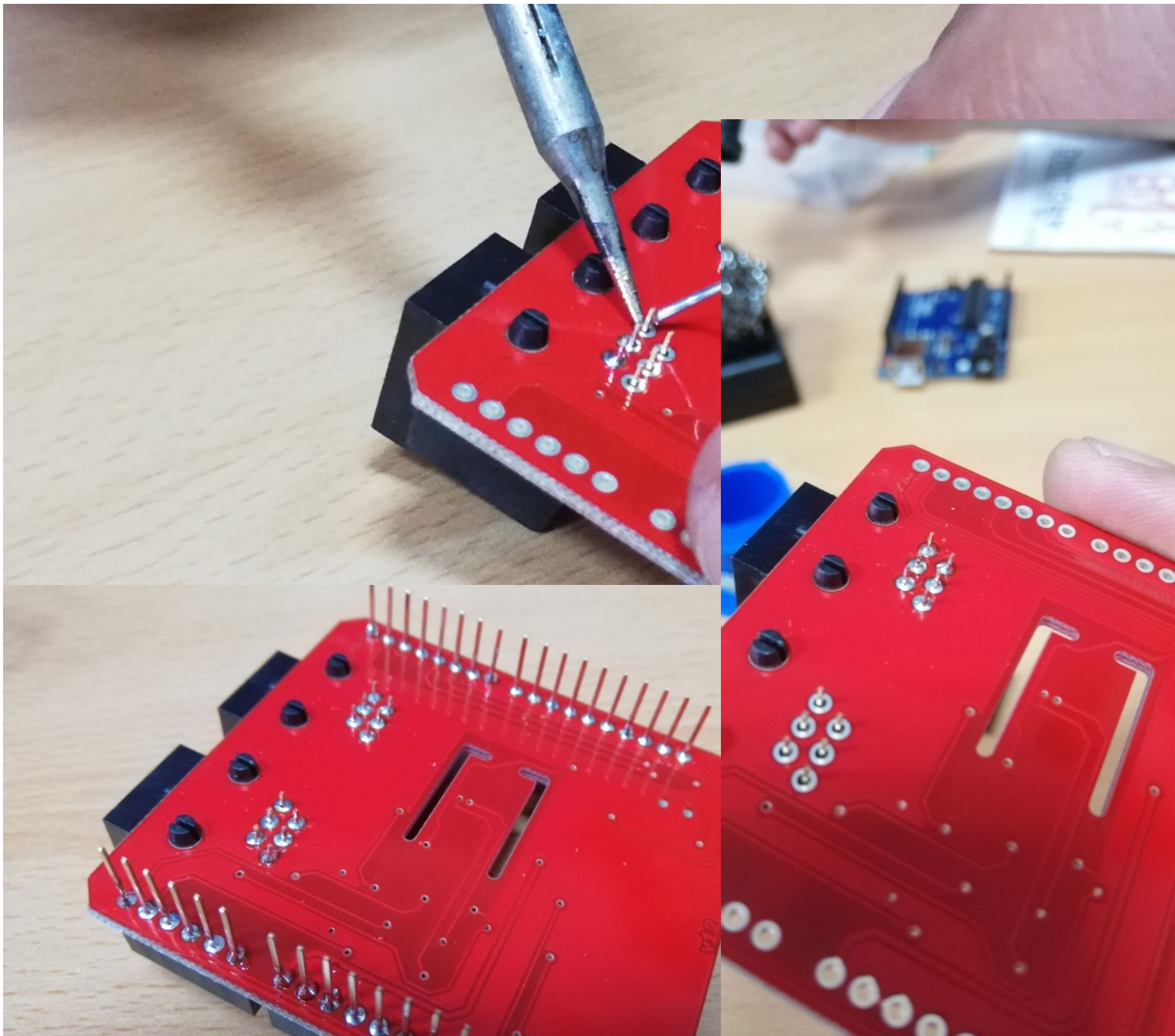
Aquí se pueden ver todas las piezas por separado, Weather Shield, Arduino, 2 conectores RJ-11, BME-280 sin soldar (a la izquierda debajo) y patillas para soldar.

Si no sabes soldar con estaño, puedes aprender como en este video:

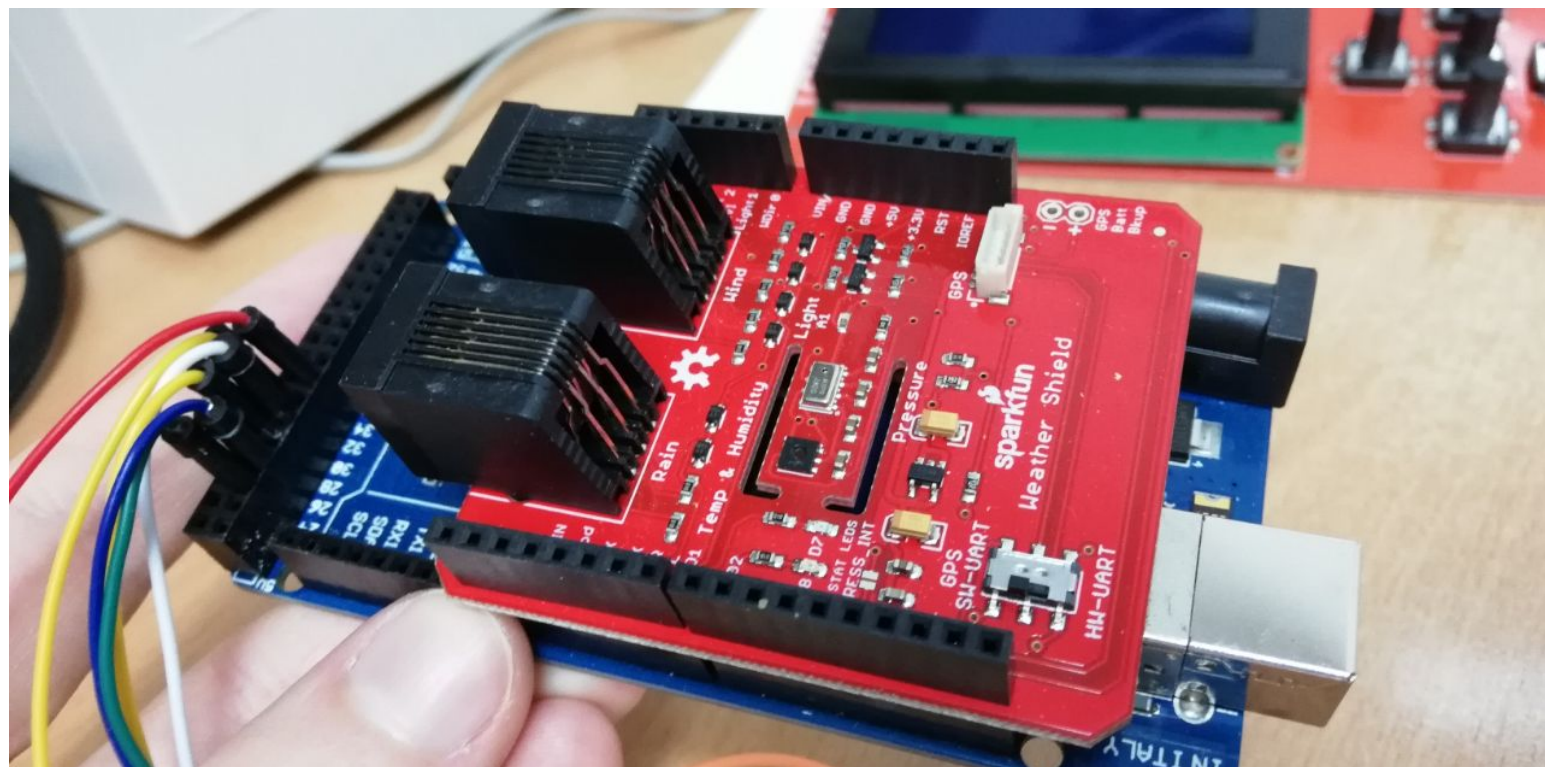
A partir del minuto 10:15 para soldar pines, aunque todo el video es interesante.

<https://www.youtube.com/watch?v=fDbG3QGN-ts>

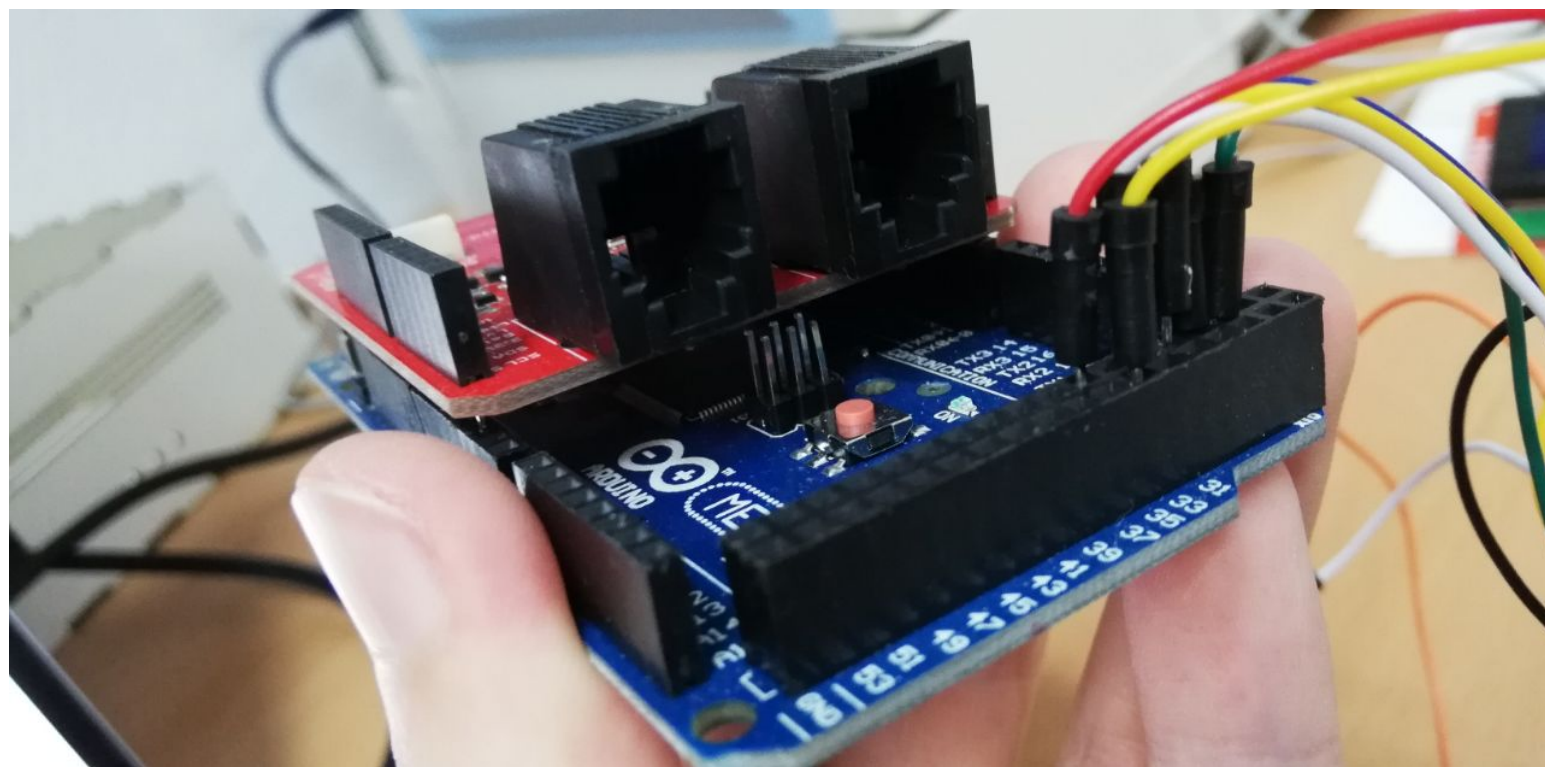
[https://www.youtube.com/watch?v=y\\_Q2kFfXxUY](https://www.youtube.com/watch?v=y_Q2kFfXxUY)







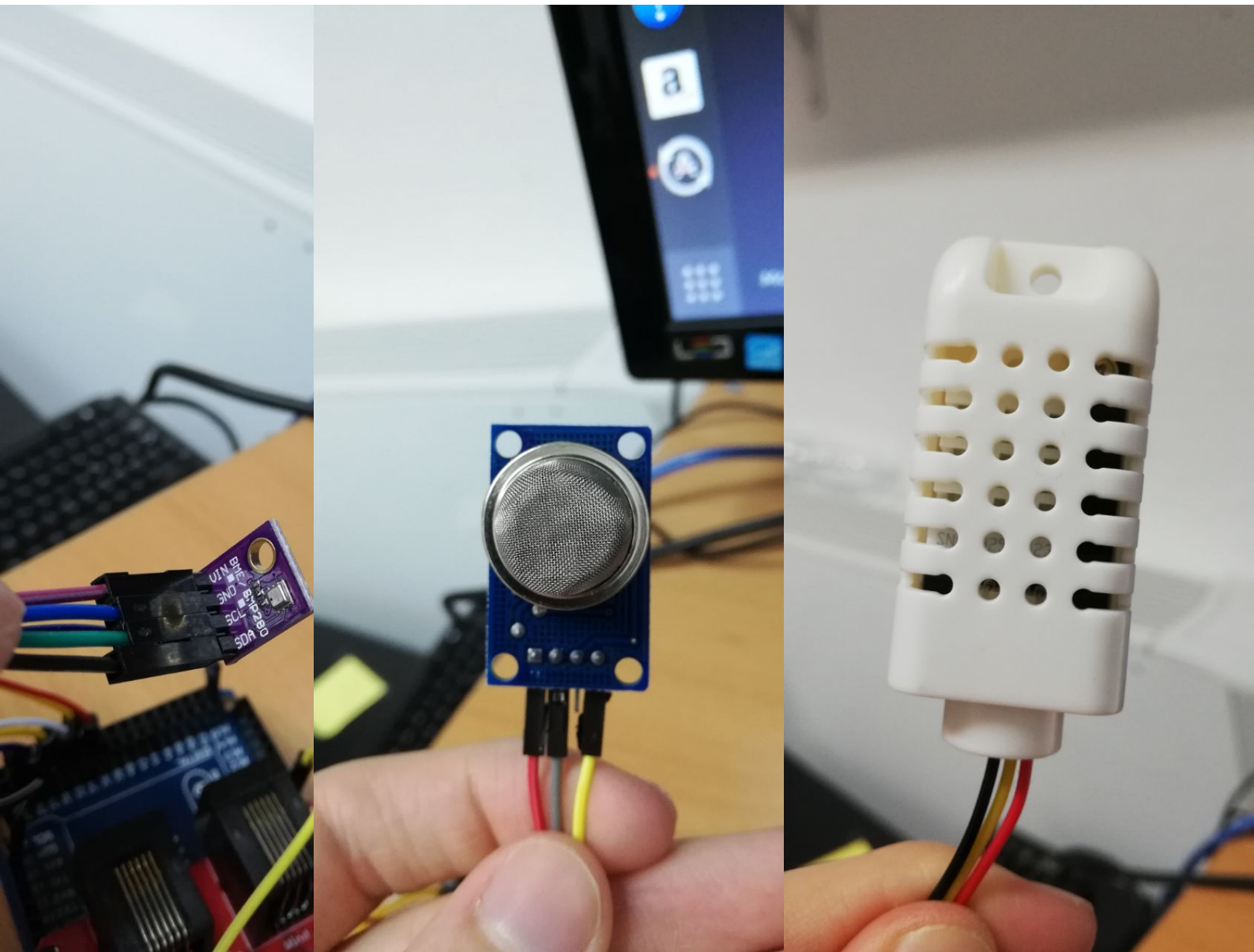
Todo ensamblado, los pines de arduino ahora están expandidos arriba, aunque no todos estarán disponibles puesto que el weather shield usa muchos de ellos para sus propios sensores.



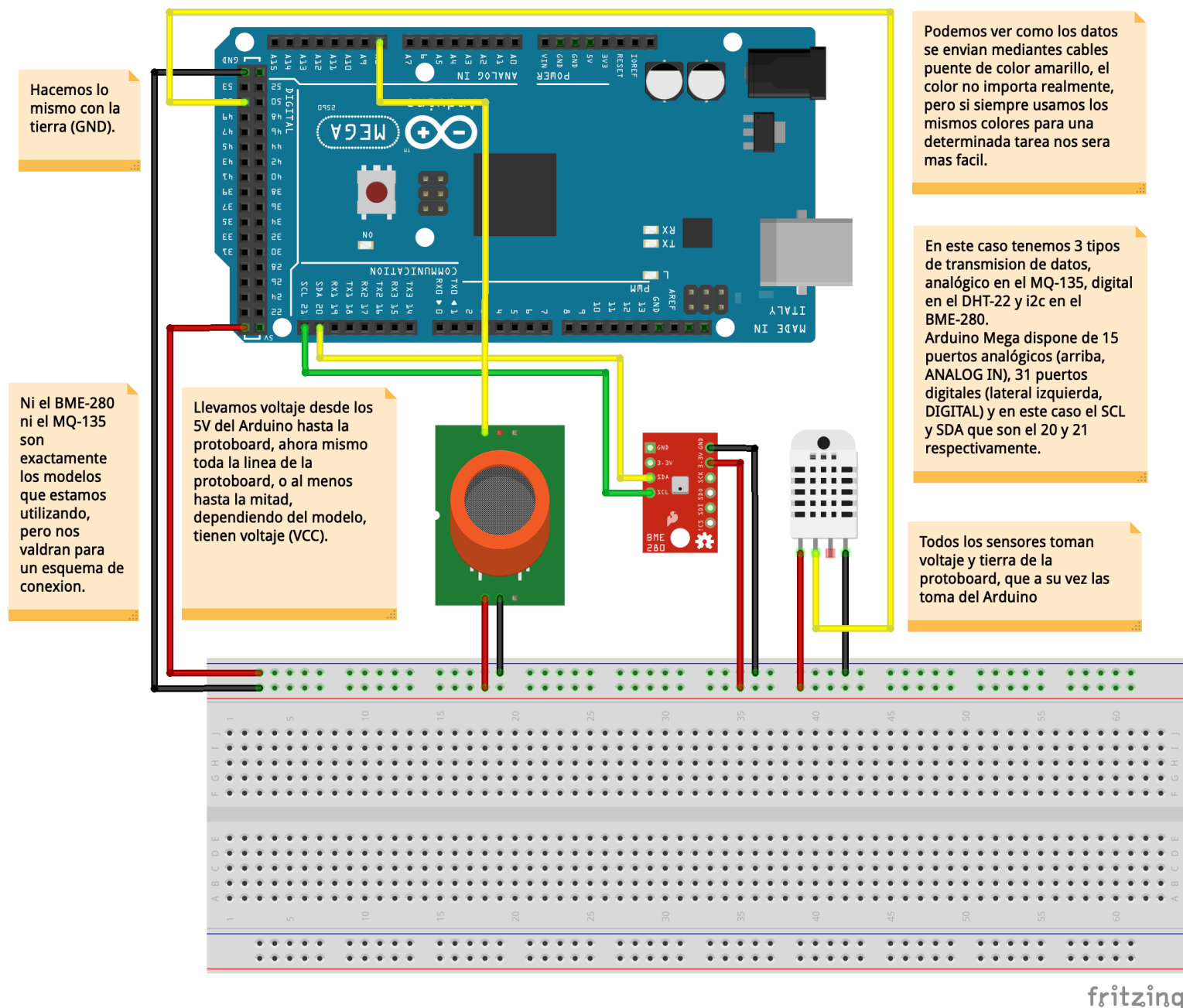
Al igual que en el caso anterior, también han de soldarse los pines del BME-280, los otros dos sensores no necesitan ninguna modificación, procedemos a conectarles cables puente para alimentarlos y que puedan transmitir datos al Arduino.

Se suelen conectar rojo a voltaje y gris/negro a tierra, en este caso el MQ-135 y DHT-22 tienen sus datos conectados en amarillo.

El caso del BME-280 es distinto ya que usa i2c, por lo tanto sus cables de datos van a SDA y SCL, pin 20 y 21 en Arduino Mega.







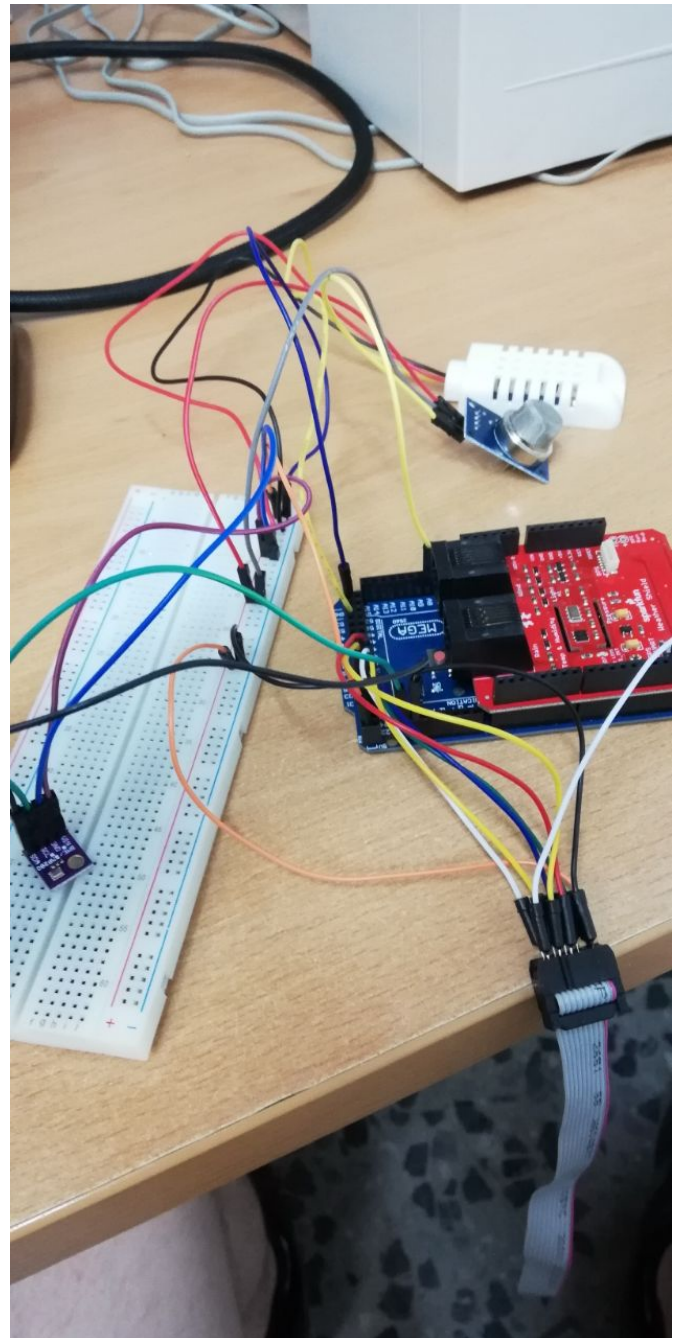
Esquema de conexión de todos los sensores, en este caso no he puesto el shield para hacerlo más claro, sin embargo estaría encima del Arduino.

En este punto, con todos los sensores conectados y el shield acoplado ya se puede ensamblar el anemómetro, pluviómetro y veleta, cuyos datos serán enviados al weather shield a través de sus conectores RJ-11.



Todos los componentes ensamblados y conectados, el cable extra es de la pantalla lcd, si has seguido la recomendación de usar un lcd i2c para Arduino tendrás que conectar menos cables, sino se ocuparán casi todos los pines digitales laterales para conectarla.

Una vez hemos acabado aquí, podemos probar que todo esta correcto usando estos sketch (programas).



[https://learn.sparkfun.com/tutorials/arduino-weather-shield-hookup-guide-v12?\\_ga=2.79885844.2077376877.1566380492-473072835.1541776220](https://learn.sparkfun.com/tutorials/arduino-weather-shield-hookup-guide-v12?_ga=2.79885844.2077376877.1566380492-473072835.1541776220)

En este tutorial esta disponible el código de prueba tanto del weather shield con el weather meter como por separado.

Después de comprobar que todo esta en orden, podemos montar el anemómetro, pluviómetro y veleta.

El tutorial oficial de SparkFun es muy completo:

[https://learn.sparkfun.com/tutorials/weather-meter-hookup-guide?\\_ga=2.79420951.2077376877.1566380492-473072835.1541776220](https://learn.sparkfun.com/tutorials/weather-meter-hookup-guide?_ga=2.79420951.2077376877.1566380492-473072835.1541776220)

Conjunto ya montado (weather meter), un cable RJ-11 llevará la información de la veleta y el anemómetro y el otro la del pluviómetro.



## Códigos de prueba de sensores ajenos a SparkFun

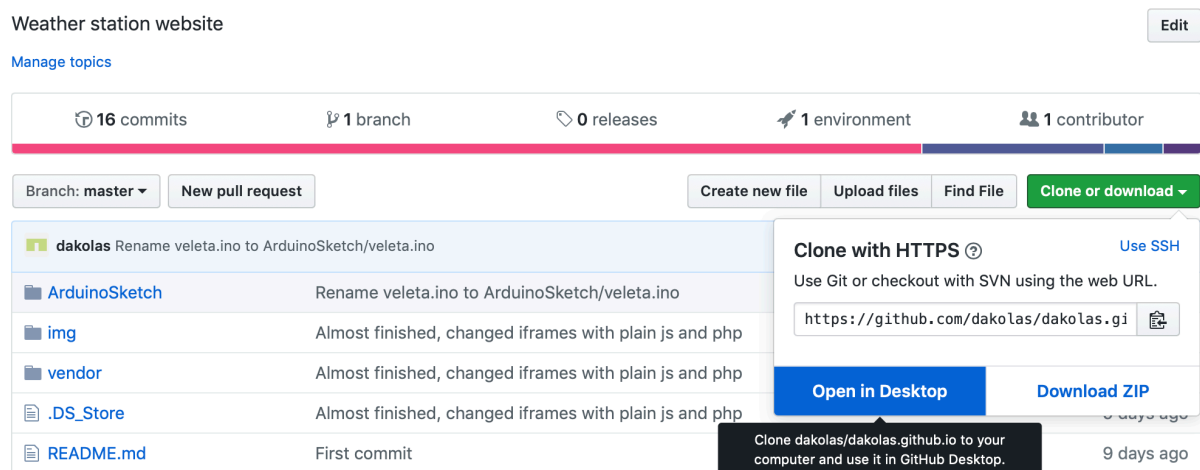
En el caso del DHT-22, MQ-135 y BME-280 se realizan pruebas por separado con librerías propias.

Una librería es un conjunto de utilidades y/o facilidades que nos aporta otra persona o compañía y que nos facilitará mucho las cosas.

Estas librerías han de ser descargadas e incluidas en el sketch de Arduino, en este enlace están tanto las librerías como el sketch (veleta.ino)

<https://github.com/dakolas/dakolas.github.io>

No se puede descargar una carpeta sola, así que descargaremos todo el proyecto pulsando en el botón verde “Clone or download” y descargándolo en ZIP (Download ZIP).



Por ahora nos interesa solo la carpeta ArduinoSketch.

## Paso 3 - Raspberry Pi

La Raspberry Pi será la puerta a internet de nuestra estación meteorológica, recibiendo los datos de los sensores desde Arduino y almacenándolos en una base de datos.

Si no estas familiarizado con el concepto de base de datos, aquí van dos enlaces que lo explican de forma sencilla.

<http://www.maestrosdelweb.com/que-son-las-bases-de-datos/>

[https://www.youtube.com/watch?time\\_continue=34&v=gfT7EGibry0](https://www.youtube.com/watch?time_continue=34&v=gfT7EGibry0) => En inglés, aunque sencillo de seguir, muy gráfico.

Almacenar nuestros datos en una base de datos nos permite no solo consultar los últimos registros obtenidos por los sensores para mostrarlos sino además disponer de ellos en el futuro creando un historial sobre el que se pueden realizar estadísticas como la temperatura media o precipitaciones de un mes en concreto, muy útil para crear un climograma.

Para configurar nuestra Raspberry Pi necesitaremos:

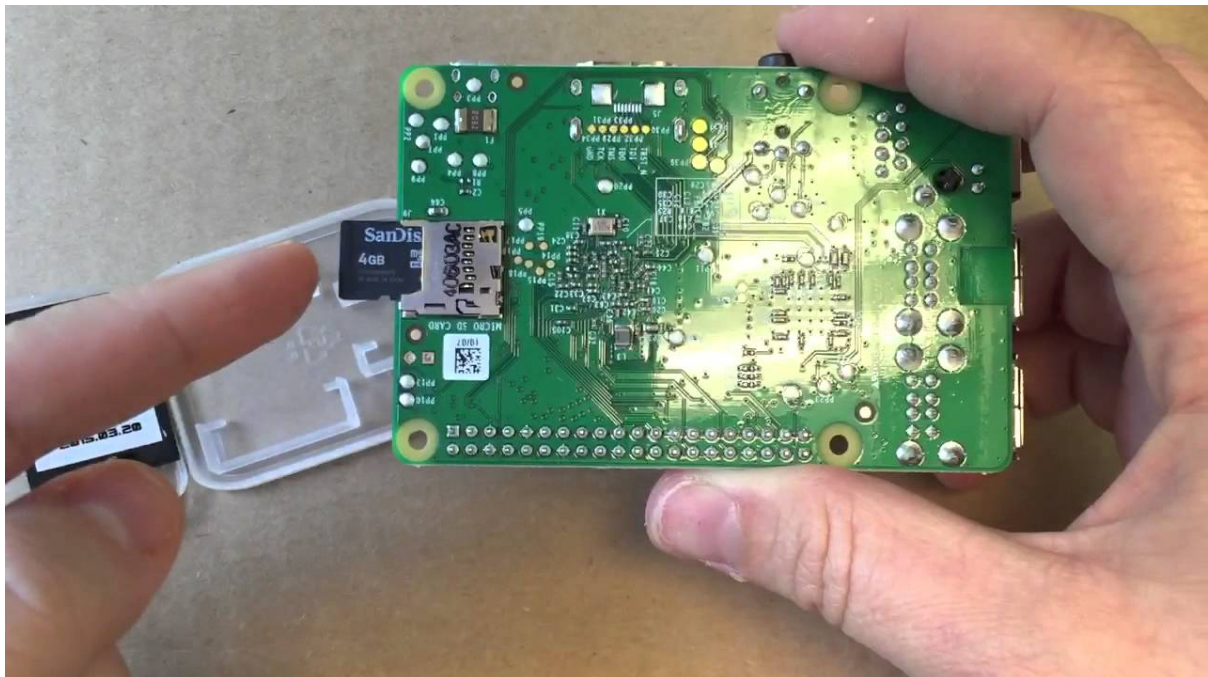
- Raspberry Pi 3
- Micro SD de al menos 16 Gb
- Noobs (Instalador sencillo del S.O Raspbian)
- Teclado y raton USB
- Pantalla y cable HDMI

Conectamos la Raspberry Pi a la corriente con su fuente de alimentación, después conectamos el ratón, teclado y pantalla.

Una vez todo esta conectado podemos empezar la instalación.

Podemos descargar Noobs aquí:

<https://projects.raspberrypi.org/en/projects/noobs-install>



Localización de la ranura para la MicroSD.

Y seguir el tutorial hasta terminar la instalación.

Si todo ha ido bien el S.O estará instalado correctamente y listo para usarse.

Veremos el escritorio, y dispondremos de una variedad de software reinstalado, entre los que se encuentra Python.

Si no sabes que es Python o un lenguaje de programación en general puedes leer los siguientes enlaces:

### **- ¿ Qué es un lenguaje de programación ?**

[https://es.wikipedia.org/wiki/Lenguaje\\_de\\_programación](https://es.wikipedia.org/wiki/Lenguaje_de_programación)

En inglés, pero el concepto se explica de forma más simple.

[https://simple.wikipedia.org/wiki/Programming\\_language](https://simple.wikipedia.org/wiki/Programming_language)

Incluso estos enlaces pueden resultar algo complejos para alguien sin base previa, y es que la definición es compleja, pero para hacerlo más simple digamos que un lenguaje de programación, especialmente de alto nivel como los que vamos a usar es una forma sencilla de decirlo al PC que queremos que haga usando comandos e instrucciones escritas en lenguaje que los humanos podemos entender, el propio PC por dentro se encarga de pasar estas instrucciones a binario para poder “entender” lo que le estamos “diciendo”.



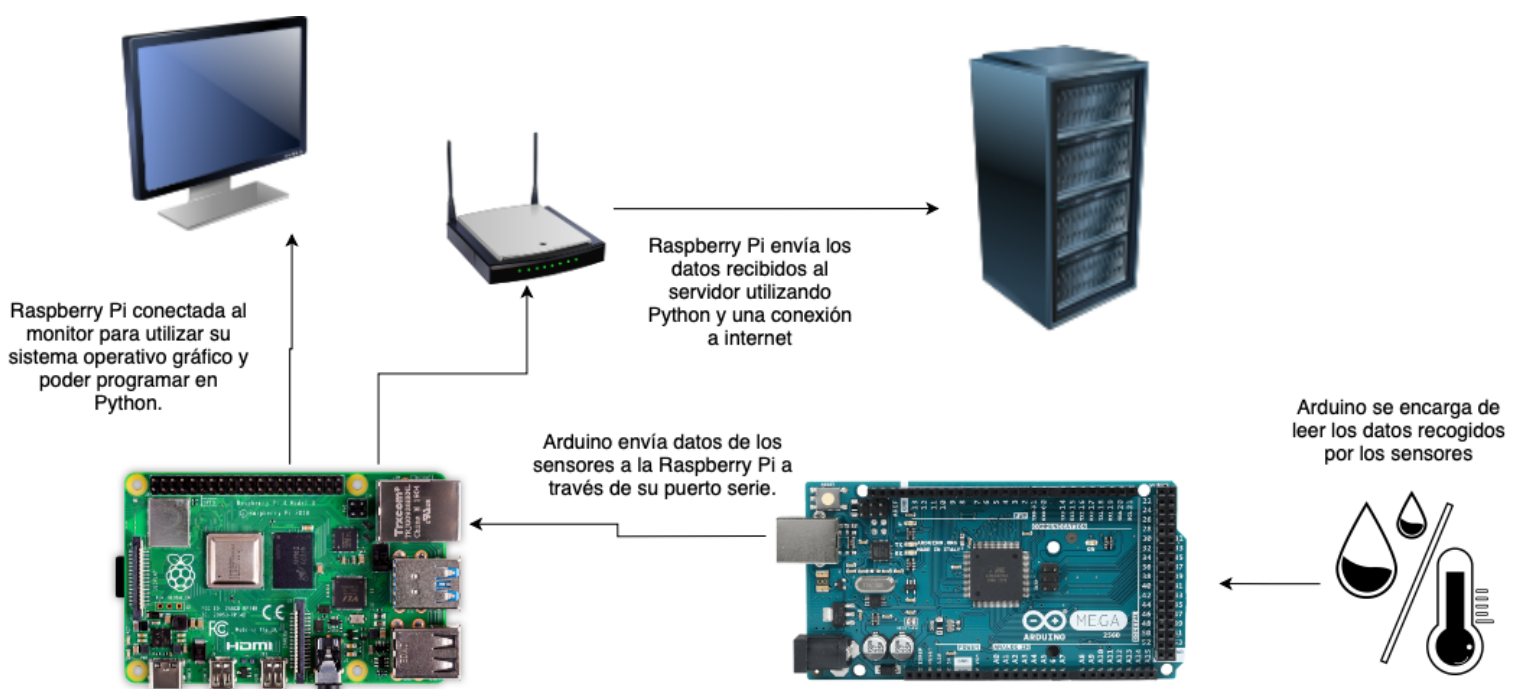
## - ¿ Qué es Python ?

<https://desarrolloweb.com/articulos/1325.php>

Después de lo explicado anterior podríamos decir que Python es un lenguaje de muy alto nivel, incluso una persona sin demasiados conocimientos podría intuir que hace un trozo de código en Python pues es más parecido al inglés que otros lenguajes.

Su sintaxis sencilla y gran comunidad lo hacen muy atractivo para empezar a aprender a programar.

En Python creamos un programa que recibirá a través del puerto serie los datos de los sensores, los guardara y los escribirá en una base de datos denominada InfluxDB.



Nuestra base de datos InfluxDB estará alojada en lo que llamamos servidor en el diagrama.

**No en Raspberry Pi, en un PC dedicado exclusivamente a esta tarea.** Podría ser el PC donde hemos instalado Ubuntu previamente, ya que la guía esta pensada para ese S.O.

InfluxDB es una base de datos de series temporales, lo cual quiere decir que su índice ( es decir, el criterio por el cual están ordenados los datos) es el tiempo en que se guardo el dato. Son una gran forma de crear históricos con datos que se leen de forma periódica, además son más fáciles de usar que las bases de datos tradicionales.

### **Para instalar InfluxDB:**

Abrir la terminal: Control + Alt + T

**Nota:** al contrario que otros puntos de la guía, esta no va a llevar un anexo explicando todos los comandos puesto que el uso de la terminal UNIX es un campo extremadamente amplio, no obstante la búsqueda de cualquiera de estos comandos en google arrojará mucha información al respecto.

Ponemos todo al día antes de instalar:

**sudo apt-get update**

**sudo apt-get upgrade**



Especificamos la fuente desde la cual queremos descargar, en el caso de que no entiendas que hacen estos comandos simplemente copia de uno en uno (son 3 en total) , pégalo en la terminal y ejecutalo pulsando intro.

```
curl -sL https://repos.influxdata.com/influxdb.key  
| sudo apt-key add -
```

```
source /etc/lsb-release
```

```
echo "deb https://repos.influxdata.com/$  
{DISTRIB_ID,,} ${DISTRIB_CODENAME} stable" | sudo  
tee /etc/apt/sources.list.d/influxdb.list
```

Una vez obtenidos los archivos de la fuente, procedemos a instalarlo:

```
sudo apt-get update && sudo apt-get install  
influxdb
```

Ya instalado lo iniciamos, este comando no nos va a mostrar nada en la terminal.

```
sudo service influxdb start
```

Abrimos la base de datos.

```
influx
```

Creamos una base de datos llamada statsdemo

```
CREATE DATABASE statsdemo
```

Mostramos las bases de datos, debería mostrarnos `_internal` y `statsdemo`, si es así todo esta correcto.

## SHOW DATABASES

*Select all fields and tags from a single measurement*

```
> SELECT * FROM "h2o_feet"
```

name: h2o\_feet

-----

time	level description	location	water_level
2015-08-18T00:00:00Z	below 3 feet	santa_monica	2.064
2015-08-18T00:00:00Z	between 6 and 9 feet	coyote_creek	8.12
[...]			
2015-09-18T21:36:00Z	between 3 and 6 feet	santa_monica	5.066
2015-09-18T21:42:00Z	between 3 and 6 feet	santa_monica	4.938

Ejemplo de uso de InfluxDB, la sintaxis esta basada en SQL, si no la conoces no pasa nada, se explica a continuación:

## **SELECT lo\_que\_queremos\_obtener FROM donde\_esta\_guardado**

En este caso `SELECT *` significa seleccionar todo.

`FROM "h2o_feet"` que es el nombre de la tabla, podría ser por ejemplo nuestra tabla de datos llamada "datosArduino".

Si quisiéramos seleccionar un dato en particular, como por ejemplo la temperatura, quedaría de esta manera:

## **SELECT temperatura FROM datosArduino**

A estas alturas ya tenemos instalada nuestra base de datos de series temporales, configurada nuestra Raspberry Pi y Arduino, es momento de empezar a programar en Python el paso de los datos de Arduino hacia nuestra Raspberry Pi y su posterior guardado en la base de datos que acabamos de instalar.

El programa expuesto a continuación, para ser tratado como Python por un PC debe guardarse como **.py**

Por ejemplo: **escribirEnBaseDeDatos.py**

Si bien Python es un lenguaje de programación de alto nivel y relativamente sencillo, no deja de ser complejo, se puede utilizar el código arriba dispuesto copiándolo y pegándolo pero si se quiere entender mejor todo este tutorial explica de forma sencilla lo básico de Python ( conceptos aplicables a muchos lenguajes de programación ).

## **Tutorial de Python**

<https://www.tutorialpython.com>

Para ejecutar el programa abrimos la terminal ( Control + Alt + T ) y nos dirigimos al directorio en el que lo hayamos guardado.

Por ejemplo, si lo hemos guardado en el escritorio:

**cd Escritorio/**

**o**

**cd Desktop/** Si tenemos Ubuntu en inglés.

Una vez situados, para ejecutar el programa usamos el siguiente comando:

**python nombre\_del\_programa.py**

Donde nombre\_del\_programa es el nombre con el que hayamos guardado el programa.

El programa se encuentra en el siguiente enlace.

<https://github.com/dakolas/dakolas.github.io>

Ya deberíamos haber descargado antes todo el proyecto al necesitar la carpeta ArduinoSketch ( Página 19 ), si es así lo tenemos en la localización:

/Descargas/dakolas.github.io-master/sendToInfluxDB.py

**o**

/Downloads/dakolas.github.io-master/sendToInfluxDB.py

Al ejecutarlo se debería leer en la terminal:

Starting...

Esperando para calibrar sensores... ( Tarda un minuto, necesario si se acaba de encender el sistema para evitar valores iniciales incorrectos)

El programa funciona con un bucle infinito, por tanto estará constantemente comprobando si ha recibido desde el puerto serie de Arduino la palabra “Ready”, que indica que se han enviado todos los valores, una vez recibida dicha palabra procederá a escribirla en la base de datos, esto ocurrirá cada 15 segundos, intervalo que tarda Arduino entre envío y envío.

Cada 15 segundos se mostrara en nuestra terminal la información recibida desde el puerto serie, para que podamos comprobar que estamos recibiendo y si algún dato no nos cuadra.

Este programa se ejecutará de manera ininterrumpida hasta que lo cerremos pulsando Control + C, comando para cerrar un programa en ejecución en Linux.

## Paso 4 - Página web

Para mostrar los datos de forma gráfica y accesible.

Resumamos, hasta ahora hemos:

- Obtenido los materiales
- Instalado en un PC el sistema operativo Ubuntu y el IDE de Arduino.
- Ensamblado los componentes en Arduino
- Probado su correcto funcionamiento
- Configurado Raspberry Pi
- Instalado la base de datos InfluxDB en el PC con Ubuntu
- Creado en Raspberry Pi el programa en Python dedicado a recibir de Arduino y escribir en la base de datos

Por ahora tenemos una estación meteorológica funcionando correctamente, si bien esto es un paso importante e imprescindible del proyecto, si no conseguimos que estos datos sean accesibles a la ciudadanía de forma cómoda estaremos perdiendo el mayor potencial del proyecto.

Por tanto el siguiente paso es crear una página web y una app móvil en la que se muestren los datos recogidos.

Este apartado está destinado únicamente a la página web.

Al igual que el programa en python, ya tenemos descargados todos los archivos de la página, son los siguientes:

- composer.json
- composer.lock
- index.php
- style.css
- vendor (carpeta)

Para ejecutar estos archivos necesitaremos un servidor web, **Apache** en este caso, para **instalarlo** seguir los siguientes pasos:

Actualizamos primero

- **sudo apt update**

Instalamos Apache Web Server

- **sudo apt install apache2**

Ejecutamos este comando para comprobar si nuestra salida por pantalla corresponde con la mostrada abajo.

- **sudo ufw app list**

Output

Available applications:

Apache

Apache Full

Apache Secure

OpenSSH

Dotamos de mas permisos al servidor web Apache.

- **sudo ufw allow 'Apache'**

Y verificamos que los cambios se han ejecutado correctamente, debe salir lo mismo en la terminal.

- **sudo ufw status**

Output

Status: active

To	Action	From
--	-----	----
OpenSSH	ALLOW	Anywhere
Apache	ALLOW	Anywhere
OpenSSH (v6)	ALLOW	Anywhere (v6)
Apache (v6)	ALLOW	Anywhere (v6)

Comprobamos que esté correctamente iniciado, podemos ver en la salida por pantalla que pone: “Active: active (running) since <date>”, esto quiere decir que se esta ejecutando correctamente.

- **sudo systemctl status apache2**

Output

```
• apache2.service - The Apache HTTP Server
  Loaded: loaded (/lib/systemd/system/
apache2.service; enabled; vendor preset: enabled)
  Drop-In: /lib/systemd/system/apache2.service.d
           └─apache2-systemd.conf
  Active: active (running) since Tue 2018-04-24
20:14:39 UTC; 9min ago
  Main PID: 2583 (apache2)
  Tasks: 55 (limit: 1153)
```



```
CGroup: /system.slice/apache2.service
└─2583 /usr/sbin/apache2 -k start
└─2585 /usr/sbin/apache2 -k start
└─2586 /usr/sbin/apache2 -k start
```


Por ultimo comprobaremos que esta accesible, para ello abrimos un navegador de internet cualquier y escribimos en la barra superior de búsqueda:

- **`http://nombre_del_servidor`**

En nombre del servidor escribimos “localhost”, es decir:

`http://localhost`

Esto nos debería abrir lo siguiente ==>



## Apache2 Ubuntu Default Page

**It works!**

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

### Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in** [/usr/share/doc/apache2/README.Debian.gz](#). Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```

/etc/apache2/
|-- apache2.conf
|   |-- ports.conf
|-- mods-enabled
|   |-- *.load
|   |-- *.conf
|-- conf-enabled
|   |-- *.conf
|-- sites-enabled
|   |-- *.conf

```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective `*-available/` counterparts. These should be managed by using our helpers `a2enmod`, `a2dismod`, `a2ensite`, `a2dissite`, and `a2enconf`, `a2disconf`. See their respective man pages for detailed information.
- The binary is called `apache2`. Due to the use of environment variables, in the default configuration, `apache2` needs to be started/stopped with `/etc/init.d/apache2` or `apache2ctl`. Calling `/usr/bin/apache2` directly will not work with the default configuration.

### Document Roots

By default, Ubuntu does not allow access through the web browser to *any* file apart of those located in `/var/www`, `public_html` directories (when enabled) and `/usr/share` (for web applications). If your site is using a web document root located elsewhere (such as in `/srv`) you may need to whitelist your document root directory in `/etc/apache2/apache2.conf`.

The default Ubuntu document root is `/var/www/html`. You can make your own virtual hosts under `/var/www`. This is different to previous releases which provides better security out of the box.

### Reporting Problems

Please use the `ubuntu-bug` tool to report bugs in the Apache2 package with Ubuntu. However, check **existing bug reports** before reporting a new bug.

Please report bugs specific to modules (such as PHP and others) to respective packages, not to the web server itself.

Si vemos esta página de inicio todo ha salido bien, tenemos nuestro servidor instalado y configurado correctamente.

La palabra “localhost” se refiere al propio ordenador, es decir, ejecutarlo en local.

Una vez instalado Apache, vamos a instalar PHP, el cual usamos para leer de la base de datos y “pasárselos” a JavaScript, que los mostrará en pantalla.

### **Instalación de PHP:**

Como antes actualizar Ubuntu:

```
apt-get update && apt-get upgrade
```

Después instalar PHP:

```
apt-get install php
```

Y comprobar que esta correctamente instalado:

```
php -v
```

Se mostrará por terminal una respuesta similar a esta:

```
PHP 7.2.3-1ubuntu1 (cli) (built: Mar 14 2018  
22:03:58) ( NTS )
```

Esto indica la versión instalada, por tanto la instalación es correcta.

Para ejecutar la página web debemos mover todos los archivos a una carpeta que llamaremos Web por ejemplo, la carpeta deberá contener todos los ficheros que especificamos antes:

Web (carpeta, y dentro los ficheros de abajo).

- composer.json
- composer.lock
- index.php
- style.css
- vendor (carpeta)

Moveremos la carpeta web al directorio estándar de Apache, es decir, el lugar donde Apache busca por defecto que página mostrar.

La moveremos con el comando mv:

**sudo mv ruta\_de\_lo\_que\_queremos\_mover ruta\_destino**

En este caso si hemos guardado la carpeta Web en el escritorio sería:

```
sudo mv /Desktop/Web /var/www/html
```

/var/www/html es el directorio por defecto del servidor web en Ubuntu, aquí Apache buscara un archivo index.html o index.php.

Ejecutamos la página web en el navegador:

<http://localhost>

o

<http://localhost/Web>

Y debería mostrar la página:

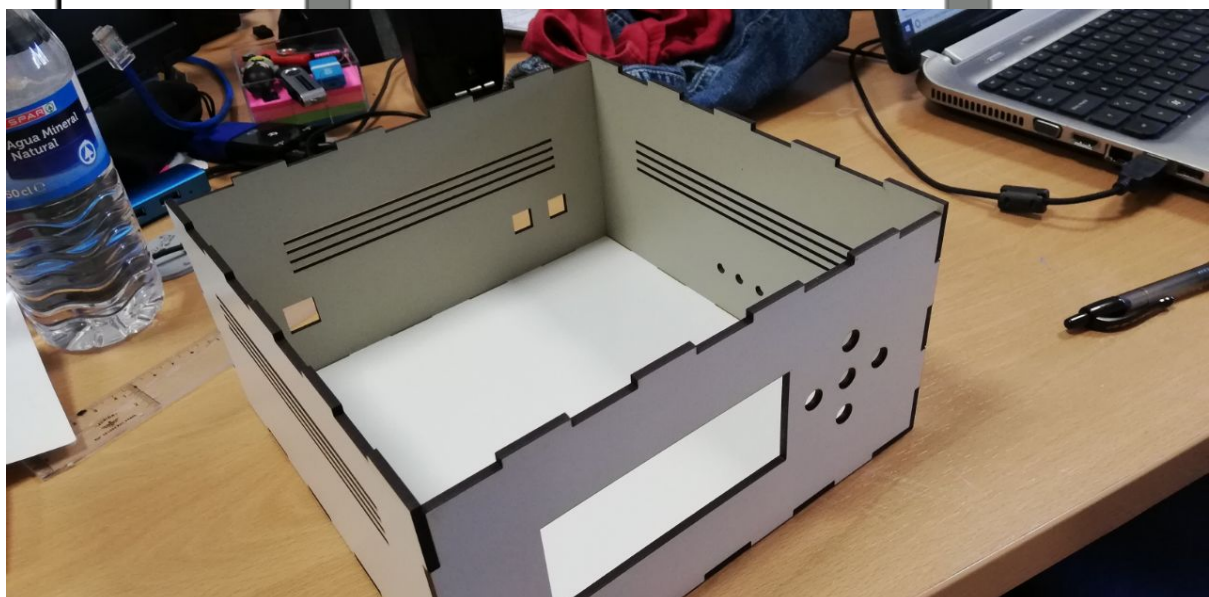
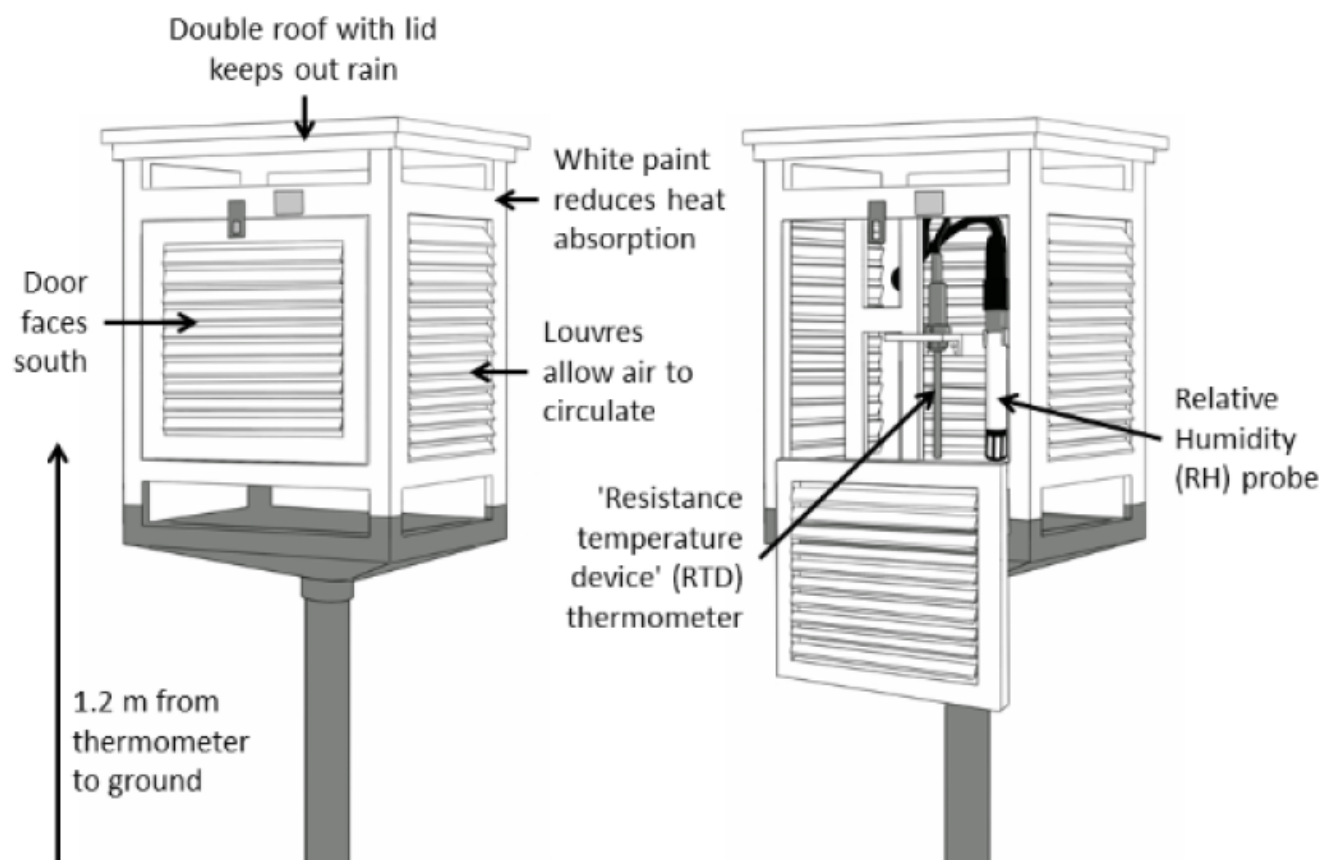
**// TODO añadir foto de la página web**

## Paso 5 - Aplicación móvil



## Paso 6 - Caja interna

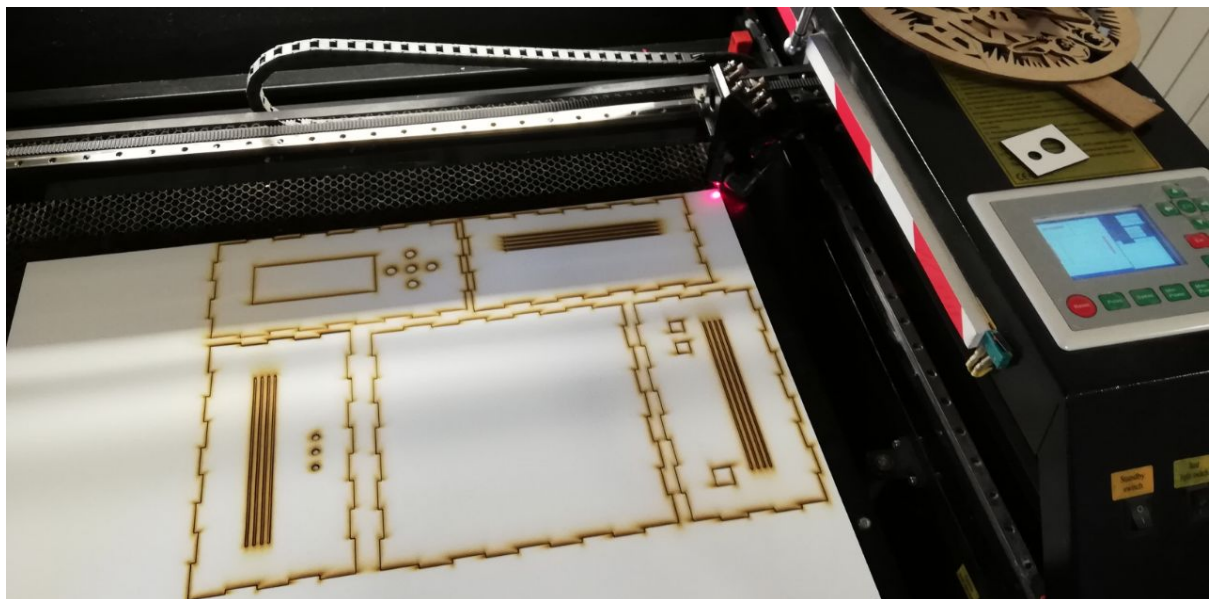
Para los componentes vamos a fabricar una caja interna que se introducirá en la pantalla de Stevenson, es decir, el armazón exterior.



Esquema de una pantalla de Stevenson, construida en madera y pintada de color blanco para absorber la menor cantidad de radiación solar posible.

Dentro de este armazón exterior vamos a construir un habitáculo interno para almacenar nuestros componentes electrónicos y protegerlos de posibles riesgos.

Para construir este armazón vamos a utilizar software de diseño para diseñar el esquema que luego cortará nuestra cortadora láser.



Aquí podemos observar el diseño recién cortado, debemos limpiar los marcas de quemado con alcohol y algodón.



Caja ya ensamblada, dispone de un frontal personalizado para nuestra pantalla, esta es la única parte que habría que cambiar si se utiliza otro modelo.

Dispone de 3 huecos para pasar 2 cables RJ-11 y 1 cable RJ-45, además de 3 aberturas laterales para los sensores que están situados en el exterior.

Para evitar sobrecalentamiento dispone de rendijas de ventilaciones en laterales y parte trasera.

### **Planos de la caja:**

**// TODO añadir links**