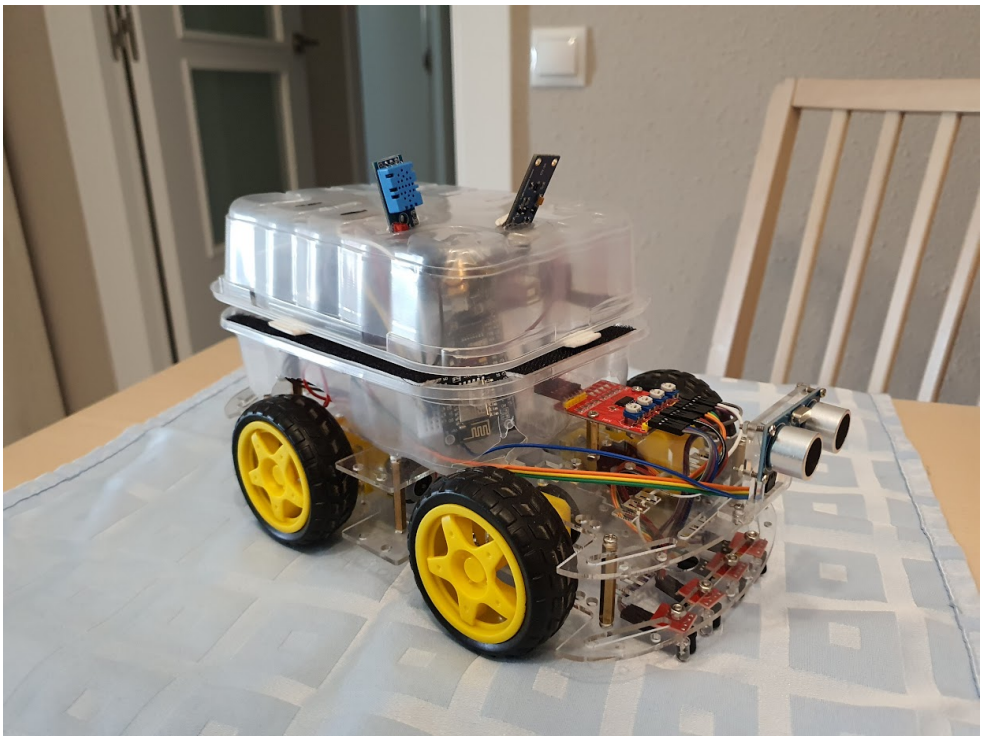
Proyecto crédito de síntesis

Vehículo de Medición Ambiental Móvil



Marko Pareja Bailén / Jaime Llastarry Jansana
Tutores: Rocco Ballester y Jordi Farrero
CCFF SMIX2
Institut Puig Castellar, a 31 de mayo de 2022

Dedicatoria.

A mi familia.

Mi familia es la brújula que guía mi camino,
mi inspiración para llegar hasta lo alto de la montaña,
el mayor consuelo cuando algo me sale mal.
Gracias familia,
por darme siempre alas para volar,
raíces para volver y
razones para quedarme.

A mis padres.

Cuando miro al cielo me invade la emoción
al saber que ya no os tengo pero estáis en mi corazón.
Por eso es muy importante que sepáis,
padres de sangre o políticos,
que no os lloro porque os recuerdo,
y que no os veo pero os tengo presentes.
Gracias por iluminar mi camino.

Índice

1 Introducción al proyecto.

¿Por qué escogimos este proyecto? Las ideas que teníamos eran individuales pero con la necesidad de hacer proyectos en parejas, elegimos la opción de fusionar los dos proyectos en uno, dando como resultado una estación meteorológica móvil, conectado vía wifi con un ordenador.

Al principio tuvimos la idea de crear una estación meteorológica y poco después tuvimos la idea de montar un coche con Arduino, y llegamos a la conclusión de hacer una estación meteorológica móvil, y el coche, además, haría cosas especiales según la información recibida desde la estación.

El proyecto consta de dos partes: una será la captación de mediciones y otra de un vehículo, que desplaza el dispositivo de captación por diferentes zonas para captar las diferentes mediciones de las mismas.

2 Qué es y en qué consiste (Introducción).

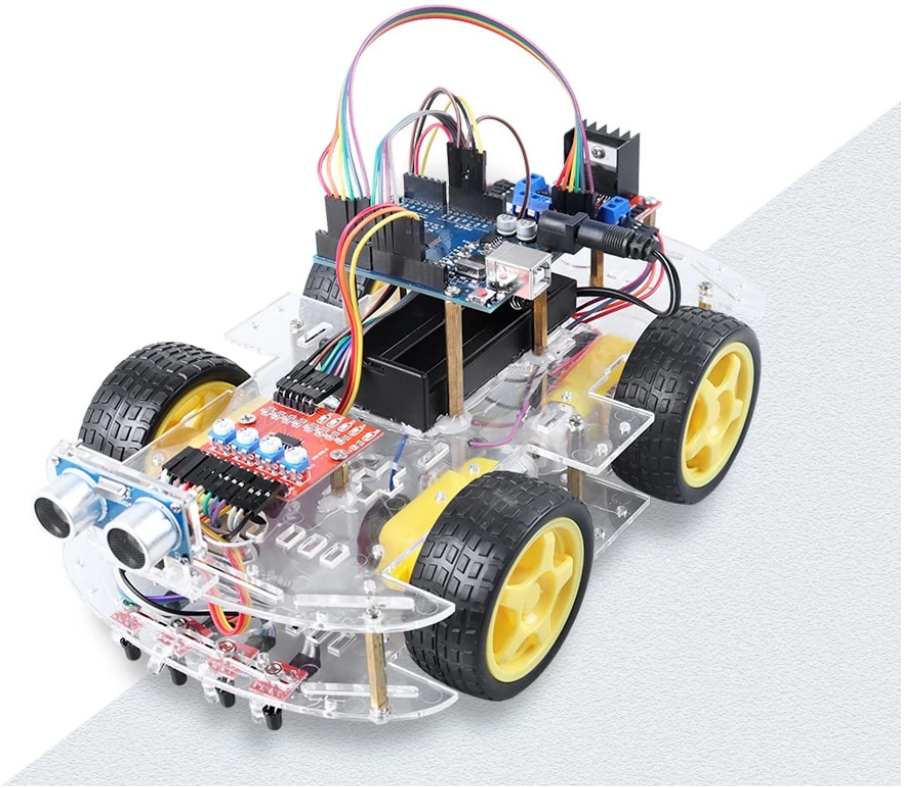
La parte de medicion (mini estación meteorológica) es montar todo en un habitáculo (tipo caja), en el cual están incluidas sus diferentes partes:

- Placa Arduino.
- Conexion Wifi.
- Diferentes sensores (humedad, presión atmosférica, temperatura, etc).
- Cableado de conexiones.
- Partes electrónicas.
- Alimentación a baterias Power-Bank.
- Una caja para incluir todo el equipamiento.

Una estación meteorológica, también conocidas como centros meteorológicos o estaciones meteorológicas domésticas, es un dispositivo que recoge datos relacionados con el tiempo y el medio ambiente, utilizando varios sensores diferentes. Dichos sensores pueden incluir un termómetro, LM35 o DHT11, para tomar lecturas de temperatura, un barómetro (BMP180) para medir la presión en la atmósfera, así como otros sensores para medir la lluvia, el viento, la humedad y otras mediciones relacionadas. Estas estaciones abarcan desde la simple tecnología analógica a la digital. Algunas, incluso se conectan a un ordenador o a Internet, por lo que los datos recogidos pueden ser analizados para hacer un seguimiento desde estación meteorológica, y a su vez, además pueden usarse para facilitar las mediciones a otras estaciones o puestos metereológicos.

La parte del vehiculo tambien consta de :

- Placa arduino.
- Cableado de cobre (conexiones).
- Ruedas, dirección, motor, chasis del vehículo.
- Sensor de proximidad y alrededores.
- Bateria.



El coche hará de medio de transporte, así podrá tomar distintas mediciones en diferentes zonas.

3 Fases de desarrollo.

Tras considerar otras opciones nos decantamos por este proyecto ya que, al hacerlo en pareja, trabajamos por separado en sus partes (dos) y, al final, reservamos una parte en común para concluir el proyecto:

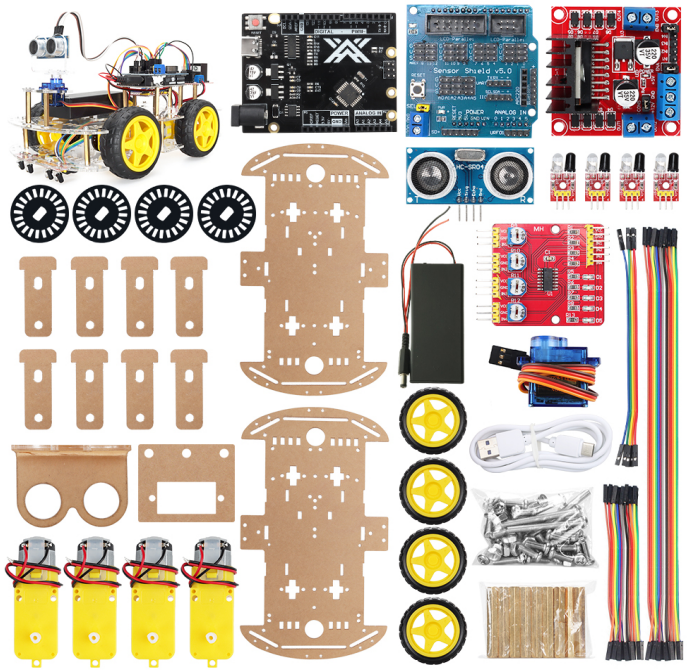
1. Buscar dónde y cómo conseguir las piezas:
   Tras ver varias opciones, nos decantamos por Aliexpress y Amazon.
2. Concretar un orden para el montaje.
3. Desarrollar un código de funcionamiento del proyecto.
4. Ponerlos a prueba por separado.
5. Ponerlos a prueba conjuntamente.

1- Adquirir piezas del montaje.

Las piezas necesarias las hemos adquirido en:

- Amazon
  - Kit medición:

- Aliexpress:
  - Robot :



Hemos cambiado algunos sensores y cables por otros, ya que nos hemos encontrado más problemas de los esperados, en el momento del montaje final.

Las herramientas utilizadas, para todo el montaje han sido:

- Cutter.
- Destornilladores: estrella y plano (pequeños).
- Alicates: planos y de corte.
- kit de soldadura.
- Cinta de autocierre.
- Blu-Tack.
- Tester.
- Tijeras.

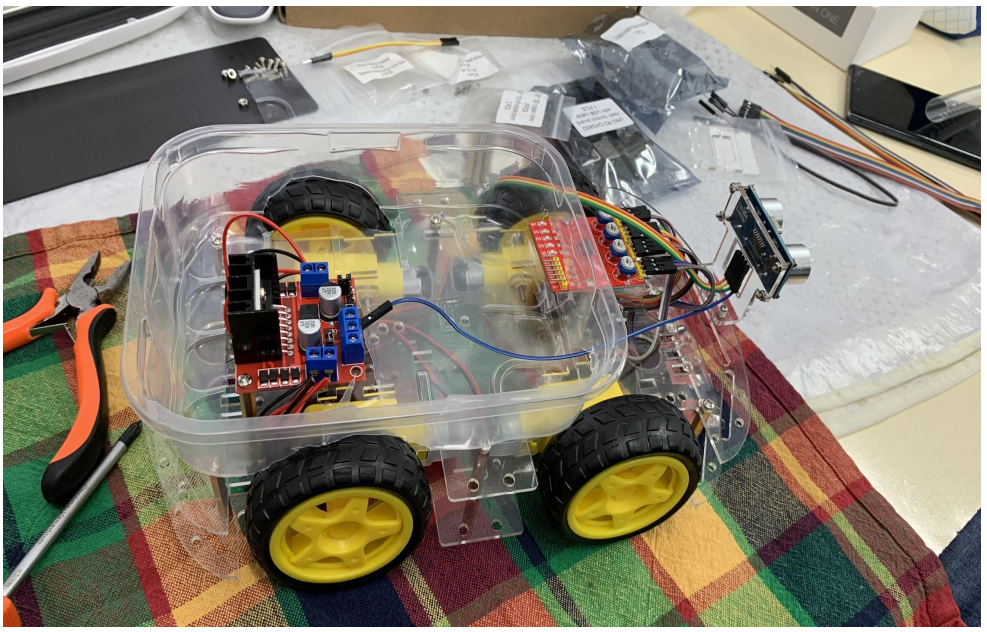
2- Orden de montaje.

En realidad, el orden del montaje en este caso, no es primordial. Sí que es cierto que sin un mínimo ya montado, parte inferior vehículo, no se puede montar el resto, pero por adelantar un poco el trabajo independientemente, hemos creído mejor ir haciéndolo así, mayoritariamente por separado.

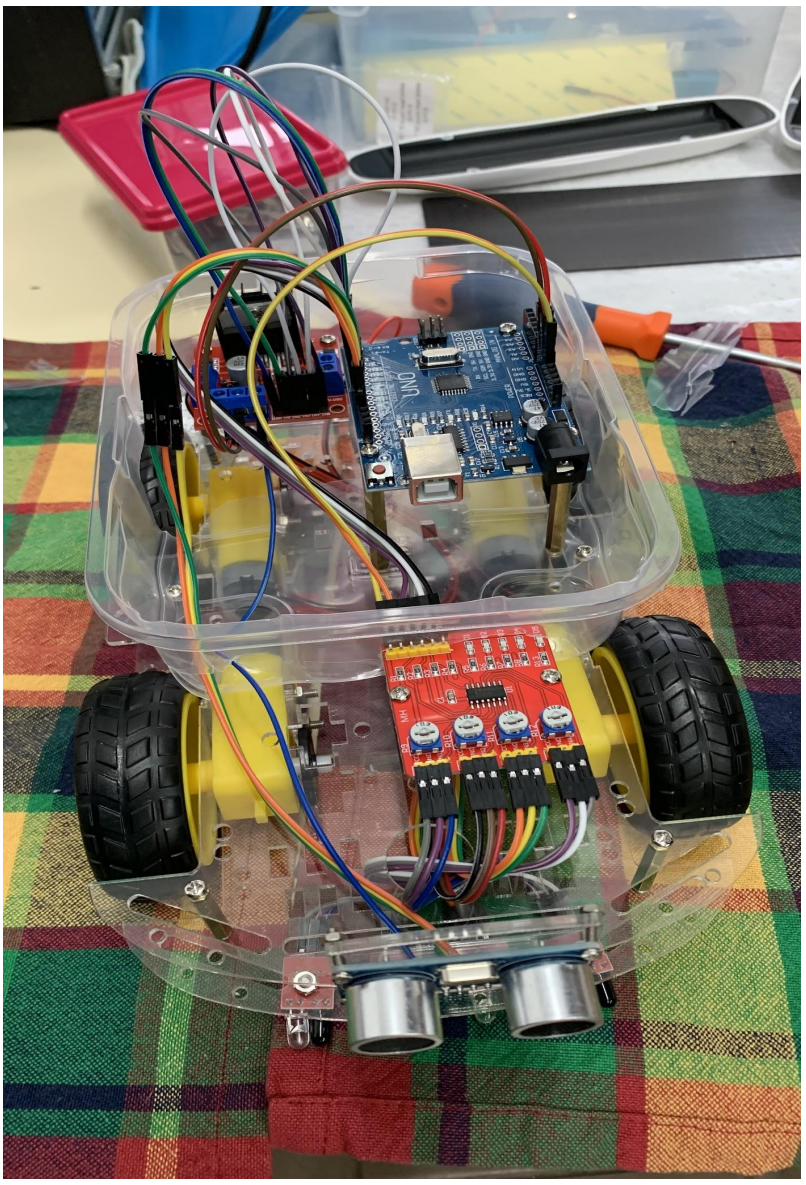
Robot:

Cuando decidí hacer un coche robot no me imaginaba todos los problemas y complejidades que habrían, el fabricante y distribuidor de el robot me paso bastantes manuales y esquemas de circuito del coche, pero el principal problema era que no había hecho eso nunca y cada esquema de circuitos era diferente, un desastre.

- Las imágenes y esquemas que se veían de exposición del producto era diferente a lo que recibimos, los agujeros donde iban los tornillos por ejemplo algunos faltaban y otros estaban mal posicionados/ diferentes a como estaban en la foto de presentación del producto.
- Primero se monta la base del coche pero tienes que tener en cuenta que hay cosas que si no montas al principio como los sensores, luego son imposibles de poner porque están en sitios inaccesibles una vez atornillar la base. El montaje en sí es sencillo pero frágil.



- Mi idea principal antes de hacer un proyecto conjunto era el coche solo, pero Jaime me propuso hacer su estación meteorológica encima de mi coche, con esta función podríamos ampliar el coche con una especie de chasis con tapers y su estación con funcionalidades extra (movimiento…).
- A la hora de montar como ya he dicho antes no está muy bien explicado ni bien indicado, entonces probando y probando conseguimos nuestro esquema de cableados y funcional, solo quedaría juntar los dos proyectos para cumplir.

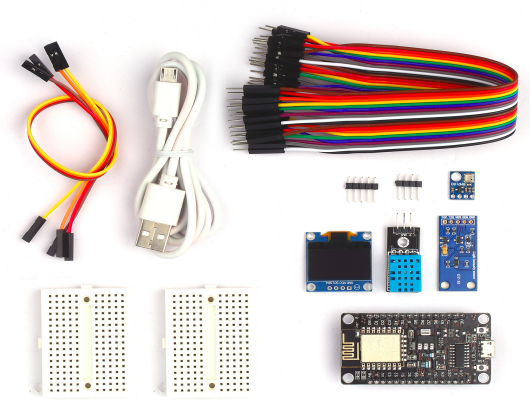


- La guía facilitada por la empresa de ventas, es eso, una guía, por lo que hemos de buscar y ajustar nuestro proyecto a ciertos parámetros; ubicación (fotografía superior), tamaño, influencia del entorno, etc.

Medidor:

En un primer momento, observo que, incluso teniendo los pasos a seguir para la construcción de la estación, no es tan fácil como parece:

- Las imágenes de muestra no son cómo las que nosotros nos encontramos; tamaño, color, medidas, etc.

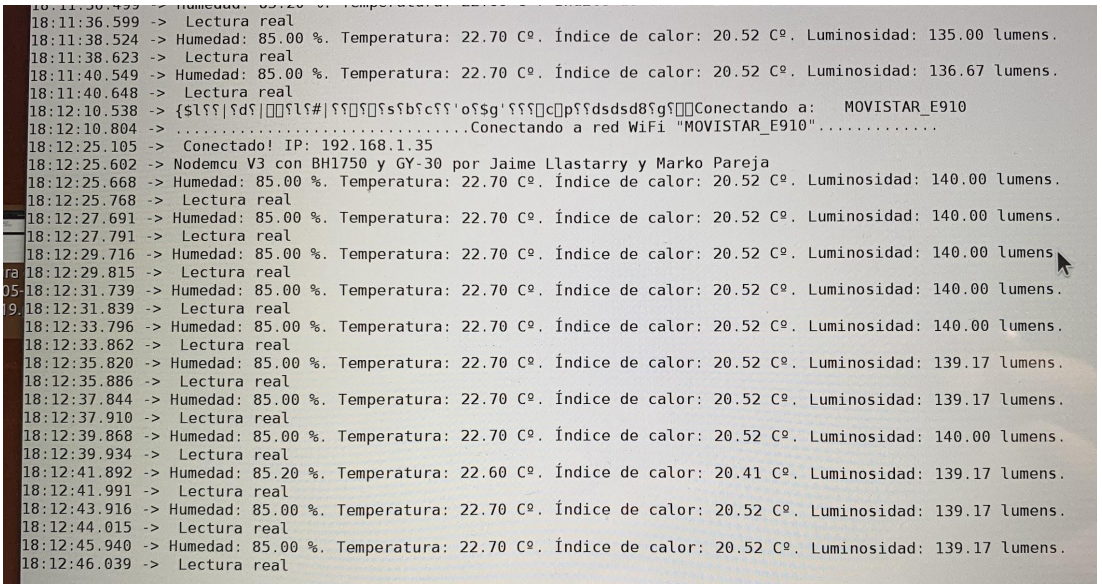


- Los parámetros de construcción son orientativos, ya que depende del "cómo" se monte, se empieza desde un punto u otro. Al unir los proyectos se pensó en incluir la parte electrónica (placa Arduino, placa ESP8266, sensores y baterías) en una zona más protegida. Se estuvo buscando recipientes, tipo caja, para añadirlo en la parte superior del vehículo; todos eran pequeños y/o pesados.
  De repente, una sorpresa, un día haciendo de comer en casa, observé unas bandejas de plástico en las que ponían la comida y, ¡sorpresa!, se me iluminó la bombilla (pensé en el proyecto). Las medí, las pesé y, tras consultarlo con mi compañero de proyecto, decidimos que era la solución que buscábamos.

3- Desarrollar código funcionamiento del proyecto.

En el momento de empezar con los códigos ninguno tenía conocimientos de programación Arduino, con lo que solo nos queda la opción de aprender sobre la marcha, y así es, en el momento que escribimos esto ya tenemos, una idea de cómo hacer los códigos y su funcionamiento: el código del coche es un poco más fácil que el de la estación, porque la estación tiene punto wifi y eso dificulta más el desarrollo del código, el código del coche tiene muchas pautas, primero declarar todos los pines del arduino y luego es poner o quitar voltaje a cada pin, y poco a poco añadir funcionalidades como por ejemplo si el sensor nota que está a menos de 10 cm de un objeto que gire… todo esto y mucho más

La estación meteorológica sigue esta pauta….. Lo principal es comprobar que los sensores funcionan, para posteriormente ir probando el código. Al no poseer ningún tipo de conocimiento (ganas sí) sobre cómo funciona la programación con Arduino, se ha ido buscando códigos para hacer las primeras pruebas: unos incluyen luces Led, otros reguladores de intensidad, etc., se parecían bastante a lo que se estaba buscando. He ido adaptando el código a nuestras necesidades, dando como resultado unas mediciones acordes con la web del centro, http://meteo.elpuig.xeill.net/, y es de una gran satisfacción, ver el resultado de ello.



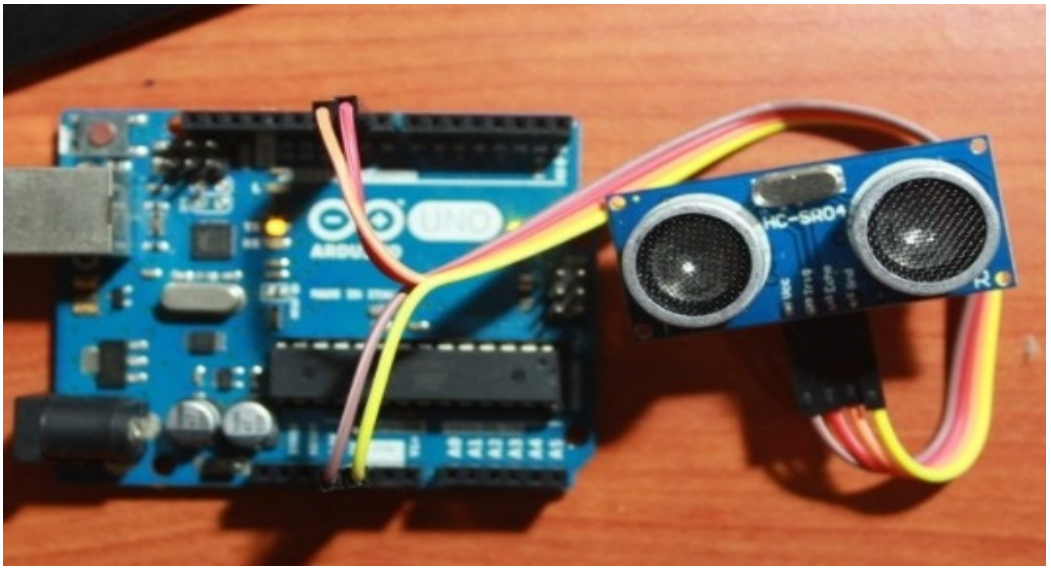
(Imagen del monitor serie del programa Arduino)

4- Ponerlos a prueba por separado.

Robot.

Antes de montar todo el coche tuve que comprobar primero que los sensores de líneas y de proximidad funcionaban correctamente al igual que los motores etc.

Sensor ultrasonidos HC-SR04



El sensor HC-SR04 es un sensor de distancia de bajo costo que utiliza ultrasonido para determinar la distancia de un objeto en un rango de 2 a 450 cm. Destaca por su pequeño tamaño, bajo consumo energético, buena precisión y excelente precio.

El sensor HC-SR04 posee dos transductores: un emisor y un receptor piezoeléctricos, además de la electrónica necesaria para su operación. El funcionamiento del sensor es el siguiente: el emisor piezoeléctrico emite 8 pulsos de ultrasonido(40KHz) luego de recibir la orden en el pin TRIG, las ondas de sonido viajan en el aire y rebota al encontrar un objeto, el sonido de rebote es detectado por el receptor piezoeléctrico, luego el pin ECHO cambia a Alto (5V) por un tiempo igual al que demoró la onda desde que fue emitida hasta que fue detectada, el tiempo del pulso ECO es medido por el microcontrolador y así se puede calcular la distancia al objeto. El funcionamiento del sensor no se ve afectado por la luz solar o material de color negro (aunque los materiales blandos acústicamente como tela o lana pueden llegar a ser difíciles de detectar).

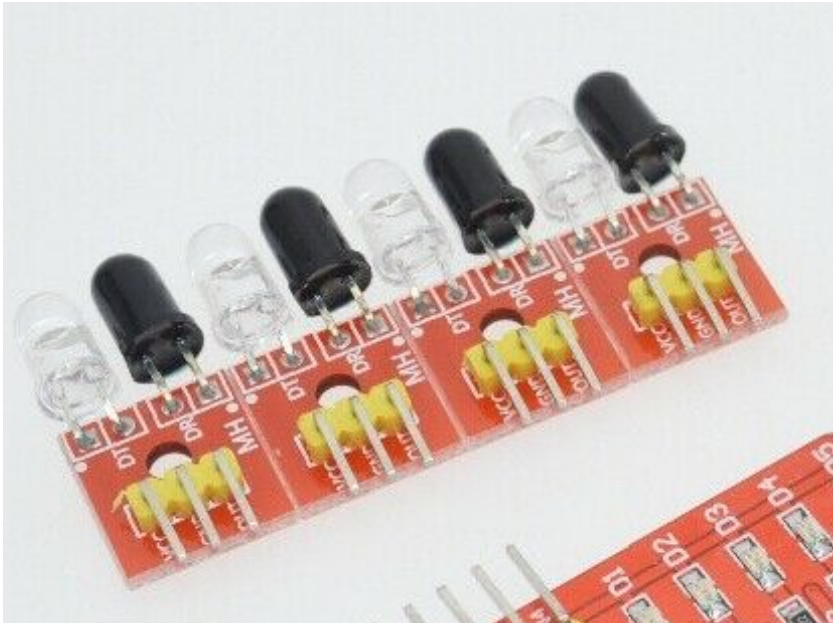
La distancia se puede calcular utilizando la siguiente fórmula:
Distancia(m) = {(Tiempo del pulso ECO) * (Velocidad del sonido=340m/s)}/2

Características de HC-SR04

- Voltaje de Operación: 5V DC
- Corriente de reposo: < 2mA
- Corriente de trabajo: 15mA
- Rango de medición: 2cm a 450cm
- Ángulo de apertura: 15°
- Frecuencia de ultrasonido: 40KHz
- Dimensiones: 45*20*15 mm

Para más informaciçon del funcionamiento práctico aquí : Codigo y pruebas con HC-SR04

Sensores de obstáculos infrarrojos FC-51



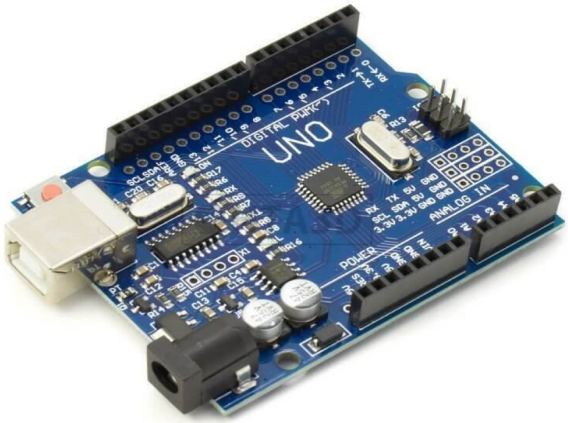
El Módulo Sensor De Obstáculos Reflectivo Infrarrojo FC-51 es un dispositivo optoelectrónico activo capaz de medir proximidad por infrarrojo IR, esta compuesto por un transmisor que emite energía infrarroja IR y un receptor que detecta la energía IR reflejada por la presencia de cualquier obstáculo en la parte frontal del módulo. El sensor puede ser usado con luz ambiente o en la oscuridad. El sensor de obstáculos reflectivo infrarrojo es usado en proyectos de robótica, que tengan como propósito el evitar obstáculos; de manera industrial para el conteo de la producción; en uso personal para sistemas de seguridad y/o detección de presencia. El módulo sensor se puede comunicar con Arduino, Raspberry Pi o cualquier microcontrolador que tenga un nivel de tensión de IO de 3.3V a 5V.

Características de FC-51

- Chip de funcionamiento: LM393
- Voltaje de alimentación: 3.3V – 5V
- Voltaje de salida digital : 5V
- Dimensiones: 31 mm x 15 mm x 7 mm
- Distancia de detección: 20 mm – 300 mm (ajustable)
- Ángulo de detección: 35°
- Pines:
    - VCC: Voltaje de alimentación
    - OUT: Salida de tensión digital (0,1)
    - GND: Tierra

Para más información aquí : Sensor infrarrojos

Placa arduino



La placa Arduino más conocida y popular. Con el microcontrolador ATmega328P SMD, 32KB de ROM y OptiBoot. 100% compatible con la original.

Arduino es un hardware open source que permite a cualquiera desarrollar proyectos electrónicos de forma fácil y sencilla. Se pueden desarrollar desde proyectos sencillos como sensores de temperatura a proyectos más completos como robots y objetos interactivos. Usando el editor Arduino IDE se puede programar el microcontrolador, y usando las entradas y salidas tanto analógicas como digitales, puedes conectar múltiples sensores, botones, diodos, displays, buzzers, etc …

La versión UNO R3 incluye algunas mejoras, como el microcontrolador ATmega328 en formato SMD a 16MHz, función autoreset, protección contra sobrecargas, conector USB para realizar la programación y nuevo bootloader OptiBoot a 115Kbps. Incluye auto selección de alimentación desde el puerto USB o desde el jack DC.

Características de placa UNO

- Microcontrolador : ATmega328P 16MHz
- Memoria : 2KB
- EEPROM : 1KB
- Interfaz : USB 2.0
- Expansión : 14 digitales, 6 analógicas, 6 PWM
- Alimentación : 7V~12V vía USB o jack DC
- Capacidad : 32 KB (512 bytes usados por el bootloader)
- Formato : Chip Atmega328P SMD

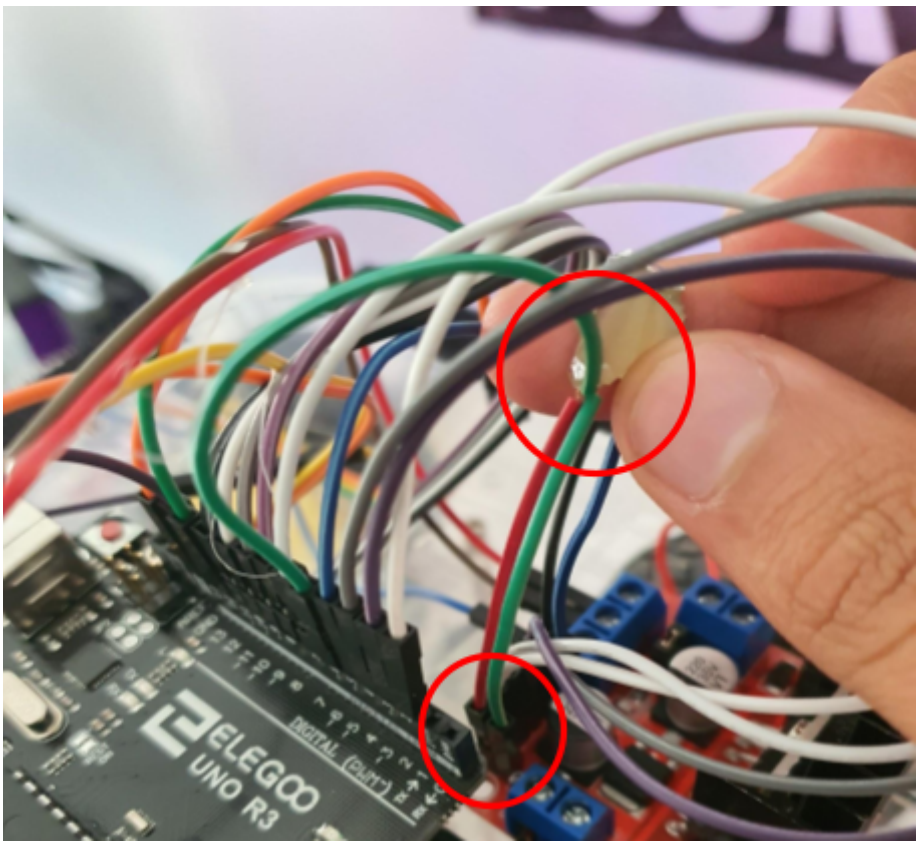
Para más información aquí : [Placa Arduino]

Pruebas iniciales robot :

Lo primero que hice al tenerlo completamente montado era comprobar su funcionamiento, probé varios códigos que habían por internet, uno me decía si el sensor ultrasónico funcionaba y a cuantos cm detectaba algo,

```
Forward
Echo =4473.00 | | Distance = 76.04cm
Forward
Echo =4452.00 | | Distance = 75.68cm
Forward
Echo =5370.00 | | Distance = 91.29cm
Forward
Echo =13497.00 | | Distance = 229.45cm
Forward
Echo =4521.00 | | Distance = 76.86cm
Forward
Echo =5393.00 | | Distance = 91.68cm
Forward
Echo =13485.00 | | Distance = 229.25cm
Forward
Echo =13530.00 | | Distance = 230.01cm
Forward
Echo =5470.00 | | Distance = 92.99cm
Forward
Echo =4647.00 | | Distance = 79.00cm
Forward
Echo =4515.00 | | Distance = 76.75cm
Forward
Echo =4526.00 | | Distance = 76.94cm
Forward
Echo =4501.00 | | Distance = 76.52cm
Forward
Echo =4550.00 | | Distance = 77.35cm
Forward
Echo =4541.00 | | Distance = 77.20cm
Forward
Echo =4523.00 | | Distance = 76.89cm
```

probe otro código que lo que hacía era encender los leds de todas las placas a ver si llega voltaje, a simple vista parecía funcionar todo pero cuanto empecé a programar el robot para que funcionara las ruedas, no iba, no iba absolutamente nada, y después de probar haciendo puentes en algunas placas me di cuenta de que si juntaba con un alambre dos pines de la placa de motores y las ruedas funcionan, ahora como podría hacer que fuera fijo, pues se me ocurrió hacer un cable que fuera un macho salida a dos hembras para así dar corriente a los dos pines y hacer un puente también,



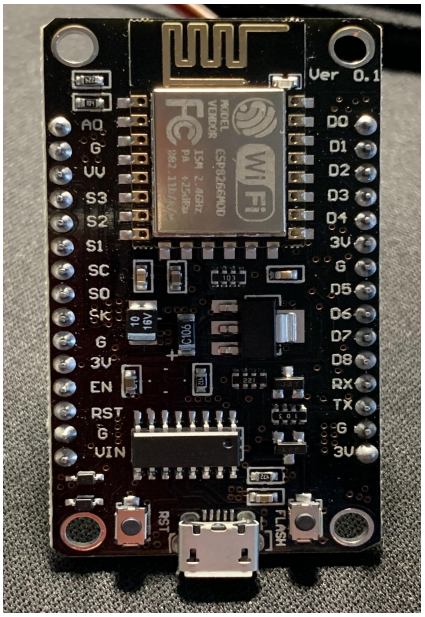
con esto ya funcionaba conectado al ordenador (video funcionando las ruedas), pero a la hora de probarlo con pilas, fallaba algo, no sabíamos lo que era pero tras probar vi que la caja de pilas transmitía el voltaje al revés, es decir, la corriente la enviaba por el tierra, lo que tuve que hacer fue cortar el cable de la caja de pilas y conectarlo al revés y ahora si funcionaba perfecto.

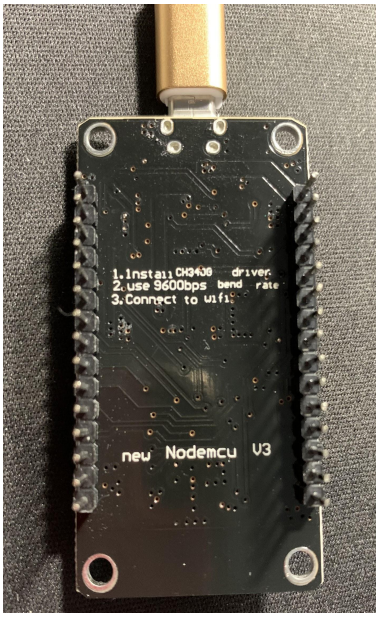
Y cuando todo parecía estar bien tras muchos días de pruebas de códigos y programación de repente las placa Arduino deja de funcionar y rechaza todos los códigos, hasta los que aceptaba antes, contacte con el proveedor, en foros, y nadie sabía nada, al final la solución me la dio un vídeo de youtube, lo que pasa es que el chip que carga los códigos se queda como pillado o algo parecido, lo que había que hacer era conectar otra placa Arduino (llamada Arduino Programar) y hacer una estructura de cableado rara y mandar un codigo que lo que hace es quemar el bootloader, básicamente como resetear el chip de arranque de códigos, como nosotros no teníamos la placa Arduino de programador me daba por perdido, pero Jaime mi compañero me recordó que él tenía un placa Arduino idéntica a la que yo usaba, que él no la necesitaba, esto nos salvó, conectemos todo y porfin 0 problemas más y todo funcionando correctamente.

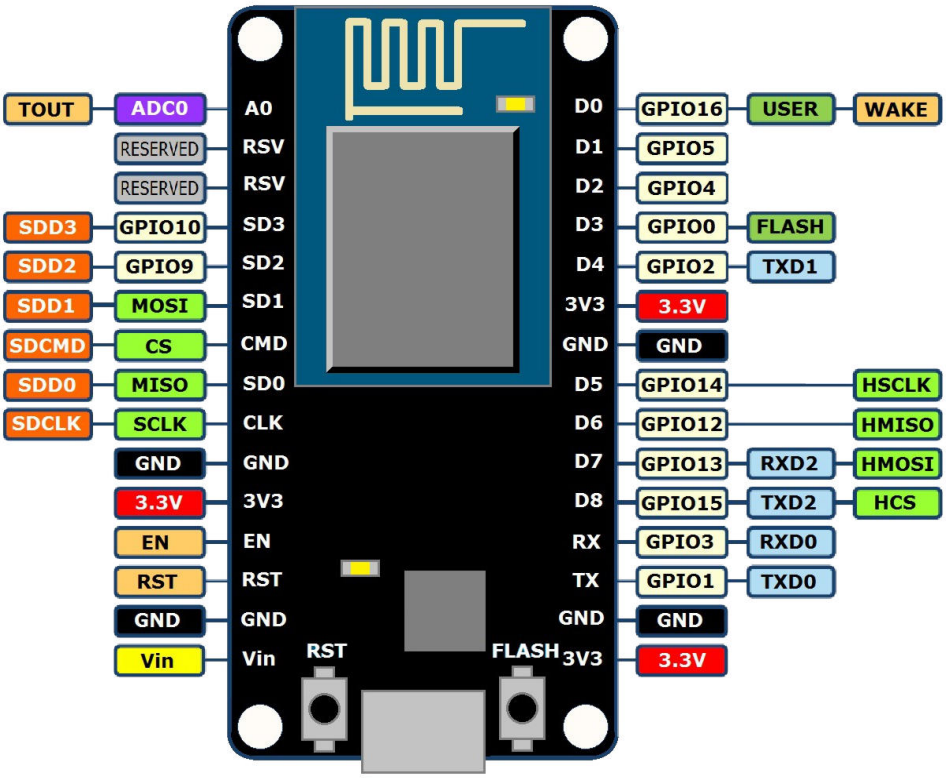
Medidor.

Se ha empezado por la parte manual, es decir, por el montaje provisional, pero primero explicamos qué es la placa ESP8266 y los sensores DH11 y el GY-30.

Placa ESP8266 NodeMCU.







NodeMCU es una placa de desarrollo con el módulo ESP12E el cuál es, seguramente, el módulo más popular que integra el SoC ESP8266. Sin embargo, pese a la popularidad de las placas NodeMCU, también existe mucha confusión respecto a la terminología (Lua, Lolin, versiones) y, en ocasiones, se confunden de versión. Pero, ¿qué es NodeMCU?

NodeMCU abarca tanto un **firmware** (en el Anexo 1: Información e Instalación programas, página 54, hay un recorte de la primera publicación sobre el firmware. La he incluido por que los datos que hay en España sobre esta publicación, no son correctos, varían en el nombre del Diario y el nombre del autor, y así, poder satisfacer mi curiosidad), Open Source como a una placa de desarrollo basados en el ESP8266.

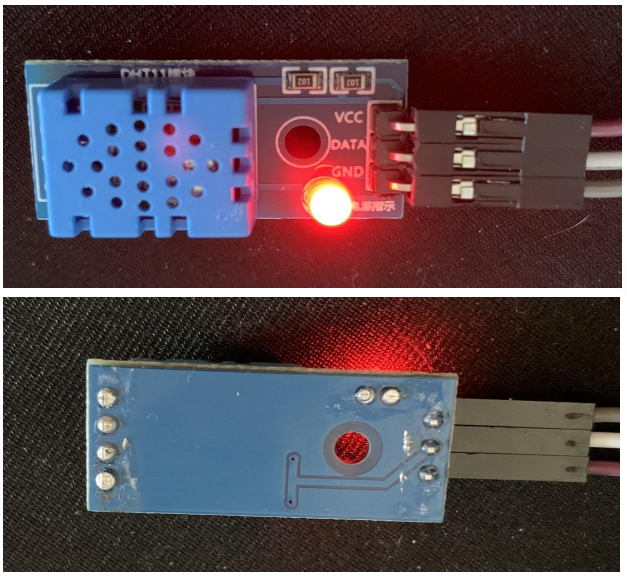
La página oficial de NodeMCU es: https://www.nodemcu.com/index_en.html

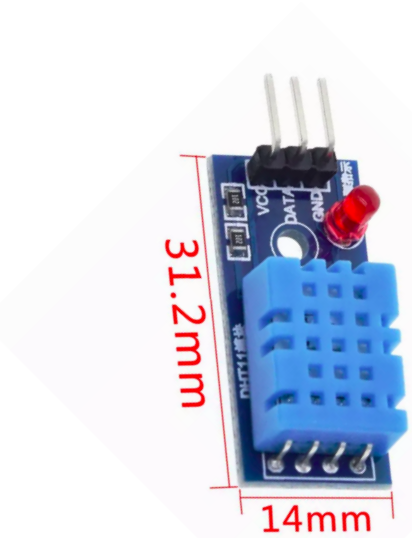
En sus comienzos, el nombre de NodeMCU era un tipo de firmware. En la actualidad yo no es así y, al hablar de NodeMCU, nos referimos a una placa de desarrollo. (Ver Anexo 1: Información e Instalacion programas)

Me decanté por éste modelo en concreto, ya que al adquirir el kit de montaje, me venía incluido. Y al ver las opciones que presentaba, conectividad y alimentación, me acabé de decidir a llevar a cabo el proyecto con esta placa.

Sensor DHT11 Digital: Humedad y temperatura.

Este sensor digital, de temperatura y humedad, DHT11 nos da una señal de salida digital calibrada, combinando temperatura y humedad. Utiliza una tecnología de captura de módulos digitales para garantizar que los productos tengan una alta confiabilidad y excelente estabilidad a largo plazo. El sensor incluye un elemento resistivo (resistencia) y un medidor de temperatura NTC, en el que va incluido un microcontrolador de 8 bits de alto rendimiento.





Características de DHT11:

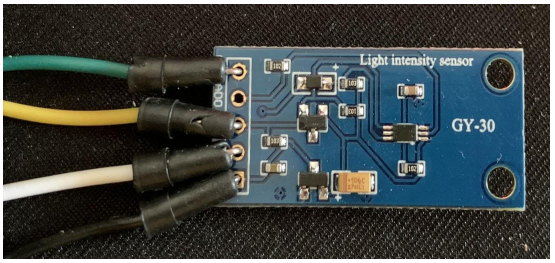
- Voltaje : 3V - 5V DC
- Rango de medición de temperatura: 0 a 50 °C
- Precisión de medición de temperatura: ±2.0 °C
- Rango de medición de humedad: 20% a 90% RH.
- Precisión de medición de humedad: 5% RH.
- Interface digital: Single-bus (bidireccional) [Es decir, un camino de comunicación entre dos o más dispositivos. Al ser un medio compartido se hace necesario un mecanismo de control para que varias señales no se solapen y distorsionen. Montado con varias líneas (anchura del bus), cada una de las cuales puede transmitir una señal binaria (un byte puede transmitirse mediante 8 líneas del bus)].
- Peso: 1 gr.
- Carcasa de plástico celeste
- Medias: 31.2mm x 14mm, aproximadamente.

Sensor GY-30 digital: Intensidad Lumínica.

BH1750FVI es un sensor de detección de luz ambiental digital, fabricado por ROHM de Japón. El convertidor, de analógico a digital de 16 bits incorporado, emite una señal digital. Esta es una versión moderna, fácil de usar, cuenta con una sencilla resistencia, que permite obtener datos válidos mediante el cálculo del voltaje. Este sensor de luz ambiental puede medir objetos directamente por fotómetro. Su unidad son los lúmenes "lux". Dicho de otra manera, los lúmenes son la medida de la cantidad total de luz visible (a simple vista) de una lámpara o fuente de luz. Cuanto más alto el número de lúmenes de las bombillas o lámparas, éstas son más «brillantes». Pongo una tabla de comparativa para hacernos una idea:

Bombillas en Watios	LED en Lúmenes	LED en Watts
25W	250lm	4-9W
40W	450lm	9-13W
60W	800lm	13-15W
75W	1110lm	18-25W
100W	1600lm	23-30W
125W	2000lm	32-40W
150W	2600lm	40-45W

Características de GY-30:



- Sensor digital del nivel de luz
- Compatible con arduino
- Modelo: GY-30
- Chip: BH1750FVI
- Comunicación. I2C [Es un puerto y protocolo de comunicación en serie, define la trama de datos y las conexiones físicas para transferir bits entre 2 dispositivos digitales].
- Medidas: 3.2 x 1.5 cm
- Fuente de alimentación 3v - 5v
- Rango de luz: 0-65535 LX

Pruebas con el medidor.

Al iniciar este proyecto no tenía ni medios, ni conocimientos para abordar esta asignatura, y mucho menos, idea de como iniciar este proyecto conjunto. La intuición y ganas de saber más, me llevaron a introducirme en el mundo de la maquinaria Arduino, que no es más que una plataforma de creación electrónica de **código abierto**, la cual está basada en hardware y software libre, y de fácil utilización para los usuarios de este tipo de plataformas.

Se empieza ingresando en foros Arduino, tutoriales, páginas web especializadas, libros, etc. Pasos para reunir las dos partes; por un lado, la idea de buen principio, una estación metereológica, y por otro lado, después de organizar en clase los grupos de proyectos, intentar fusionar dos ideas separadas en una sola. Los inicios eran los esperados, ¿cómo montamos esto?, ¿de dónde sacamos información?, ¿funcionará?. Para muestra un botón:

```
ws.on("message", m => {
  let a = m.split(" ")
  switch(a[0]){
    case "connect":
      if(a[1]){
        if(clients.has(a[1])){
          ws.send("connected");
          ws.id = a[1];
        }else{
          ws.id = a[1]
          clients.set(a[1], {client:
          ws.send("connected")
        }
      }else{
        let id = Math.random().toString(
        ws.id = id;
```

Pruebas iniciales medidor: Primeros pasos.

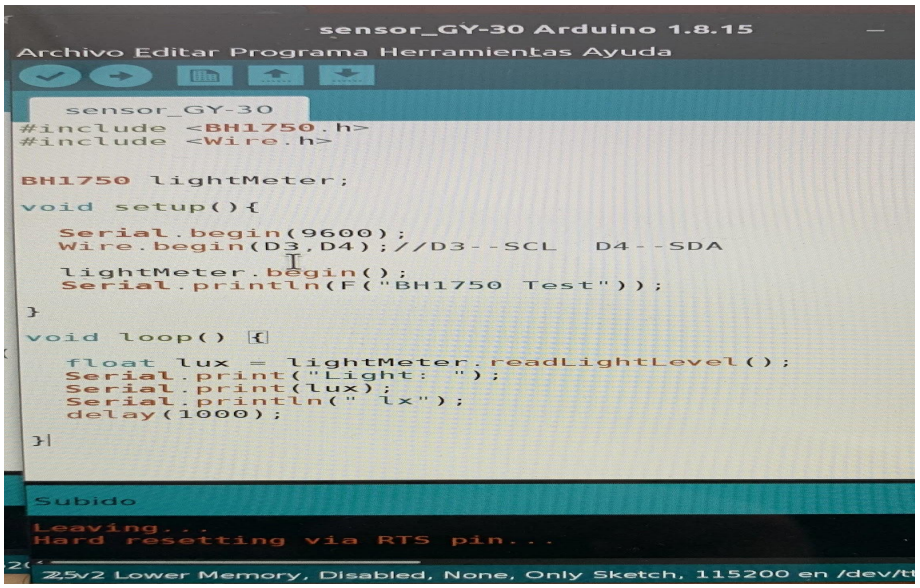
Nodemcu, como decía en el punto “**Placa ESP8266 NodeMCU**” de este proyecto, es una placa que tiene todos los elementos necesarios para conectar periféricos a las entradas y salidas de esta placa, en la cual se haya un microcontrolador y que, además, contiene una antena para poder realizar conexiones vía wifi.

En su microcontrolador se pueden grabar una serie de instrucciones, para así poder realizar circuitos eléctricos y/o electrónicos.

Utiliza el código Arduino, que es un tipo de lenguaje de programación propio, de alto nivel de procesamiento (Processing), muy similar al C + +.

Pero, ¿cómo funciona?. Sus funciones, como ocurre con casi todas las placas con microcontroladores, las podemos resumir en 2 puntos importantes:

- Posee una interfaz de entrada, la cual puede estar directamente unida a los periféricos o conectarse a ellos a través de puertos. El objetivo final de la interfaz de entrada, es trasladar la información al microcontrolador. El microcontrolador se encarga de procesar esa información. Por lo cual, se puede utilizar para infinidad de proyectos.
- Además cuenta con una interfaz de salida. Este se encarga de llevar la información procesada a los periféricos utilizados para el uso de esos datos.



Dicho esto, pasamos al desarrollo de su utilización en este proyecto.

Estructura básica de los programas Arduino.

A continuación, indico dónde dirigirse para instalar los programas necesarios para esta parte del proyecto. **Anexo 1: Instalación de Programas**.

Empezamos con una breve explicación de las partes de las que se componen los programas de este tipo, ya que sin una base, no se entiende nada del programa.

Programas.

La estructura básica de un programa Arduino es bastante simple y se compone, de al menos, dos partes. Estas partes son necesarias y en su interior hay bloques que contienen declaraciones o instrucciones. Veamos un ejemplo sencillo:

```
/* Antes de la función setup() podemos utilizar esta zona para declarar variables. Éstas se declaran mediante #:*/

// Bibliotecas: Incluyen los datos de configuración de su componente

    #include <ESP8266WiFi.h>
    #include <WiFiClient.h>
    #include <ESP8266WebServer.h>
    int led1;
/* antes de la función setup() podemos utilizar esta zona para declarar variables, cómo las Bibliotecas o int led1;/int led2;*/
    int led2;
    void setup() {
// instrucciones;  aquí realizamos la configuración inicial de variables
    }
    void loop()
    {
     // instrucciones;  el código que se va a repetir
    }
```

En la primera parte, se encuentra la zona donde se declara e inicializa las variables. Las variables son «símbolos», que son usados en programación, que guardan o almacenan valores temporales, los cuales pueden ser alterados durante la ejecución del programa. En este ejemplo, declaramos las variables **led1** y **led2**, que van a ser utilizadas para determinar las salidas digitales que vamos a utilizar en Nodemcu ESP8266 V3, y así, colocar ambos leds y poder programar su encendido y apagado. Utilizando variables nos evitamos poner el valor de esa salida digital en el código de forma constante.

- Por ejemplo, si por algún motivo deseamos cambiar esa salida digital, y conectar un led que estaba conectado a la salida o **pin** 13, a la 12, sólo tendríamos que hacer ese cambio en la declaración de la variable, y no cada vez que la utilizamos en el código.

- La función **setup()** es la parte encargada de recoger la configuración, y **loop()** es la que contiene el programa que se ejecutará en modo cíclico (de ahí el término loop –bucle). Ambas funciones son necesarias para que el programa se ejecute. La función de configuración **setup()** debe contener la declaración de las variables:

    - Es la primera función a ejecutar en el programa, se ejecuta sólo una vez, y se utiliza para configurar o inicializar las entradas/salidas digitales con la instrucción **pinMode()**, enlazar con el puerto serie y otras configuraciones iniciales.

- La función **loop()** contiene el código que se ejecutará continuamente (lectura de entradas, activación de salidas, etc). Esta función es el núcleo de todos los programas de Arduino y la que realiza la mayor parte del trabajo.

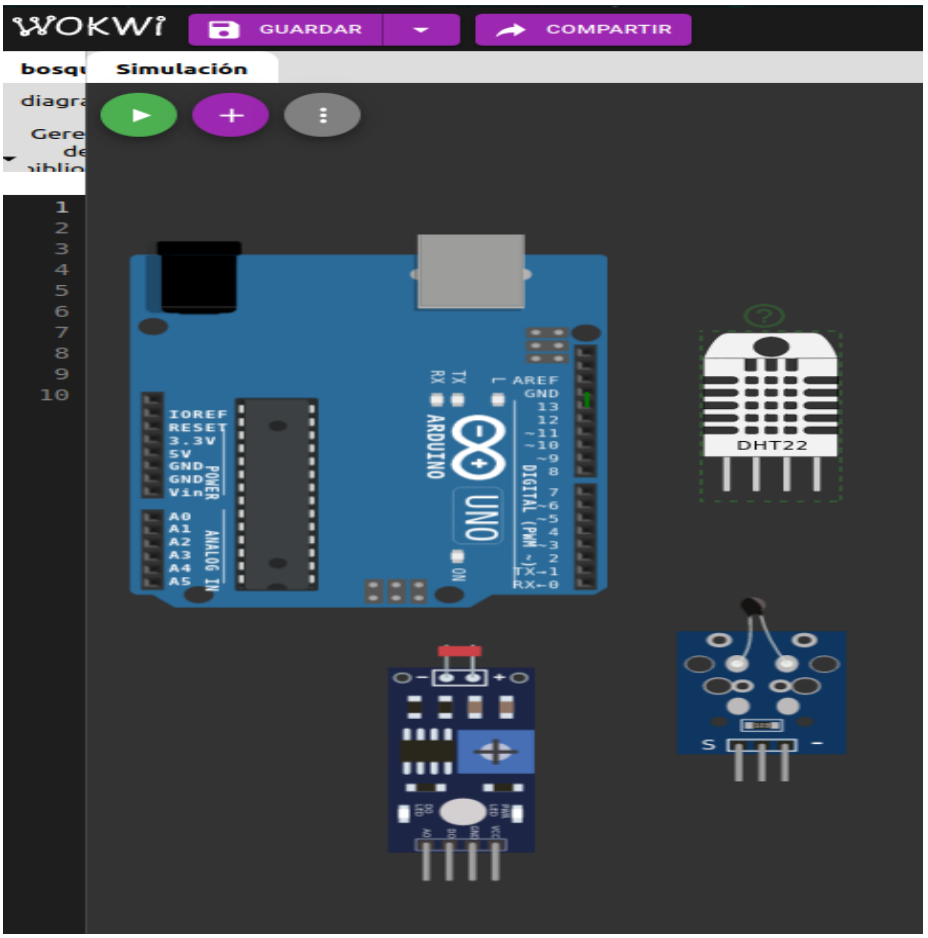
Como se observa en el bloque de código, cada instrucción acaba con "**;**" y los comentarios se indican con "**//**…línea de código**"** al principio de la línea que va a ser comentada. Igual que en el lenguaje de programación C + +, se pueden introducir bloques de comentarios con "**/*** …líneas de código ***/**", en el cual se puede escribir un comentario más extenso en varias líneas, empezando y acabando con estos símbolos.

Pero antes de ésto, hemos de tener claro qué proyecto vamos a realizar, ya que sin un proyecto en mente, no se debería de empezar a programar nada.

Después de pensar que hacer, es importante dibujar un boceto o "borrador" de lo queremos hacer. Normalmente se utiliza lápiz y papel, pero en mi caso, me ha llegado información externa (profesores y compañeros), que accediendo a una página web, "**Wokwi"**, se puede crear un boceto del proyecto y comprobar su funcionamiento, ya que permite la opción de conectar cableado y todo el programa.

Personalmente, he visto que a ésta página le faltan muchos componentes y no veo con claridad lo que me interesa pero, no está mal a grandes rasgos. La recomiendo.

A continuación, he dejado una imagen de las posibilidades que ofrece esta web con los accesorios que más se asemejan a los que he utilizado.



Para más información y posibles dudas, dirigirse a:

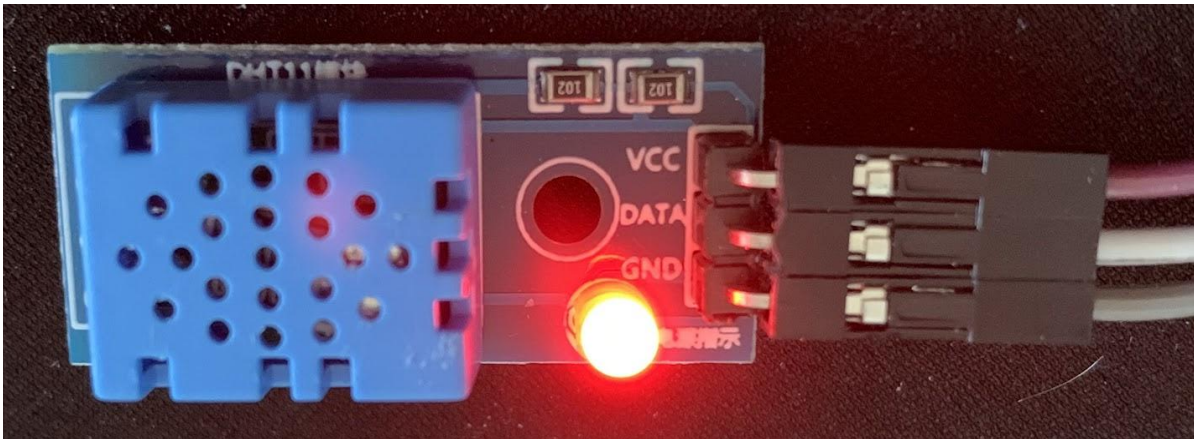
Anexo 1: Instalación de Programas.

Pruebas con el medidor: Primeras pruebas del proyecto.

Primera prueba: Sensor DHT11.

Como ya sabemos, los inicios son un poco caóticos: ¿Qué miro?, ¿Por dónde empiezo?, ¿Cómo se hace?, etc.

Empezamos por investigar el funcionamiento del sensor; que mide, cómo lo mide, alimentación eléctrica, y, lo más importante, como lo conecto con la placa Nodemcu ESP8266. Investigar en foros o páginas web, es crucial, si eres como yo, que no ha hecho nunca un proyecto así. Prosigamos. Lees en sitios de información y encuentras, muchas veces, casi todo el trabajo (casi todo) hecho: programas, esquemas y montajes. Cada uno es libre, a día de hoy, de elegir su forma de trabajar y conseguir sus objetivos. Ahora pasamos a la parte de la que trata este punto, la prueba con el sensor DHT11.

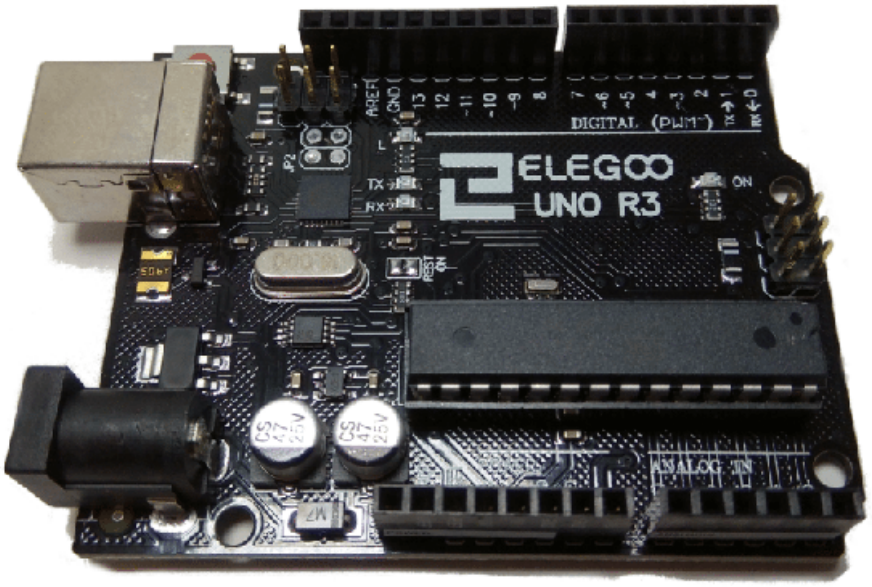


Tenemos que seguir un cierto orden si queremos que los accesorios eléctricos y electrónicos nos duren mucho tiempo. No hay que correr.

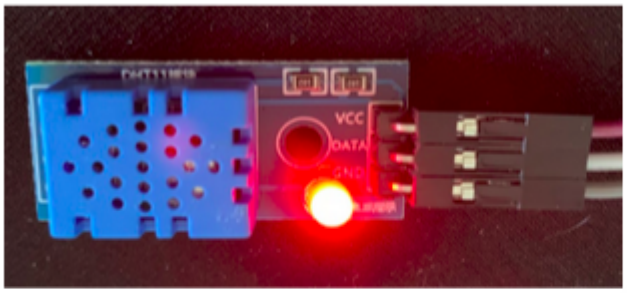
1. Miramos lo que necesitamos para las pruebas.

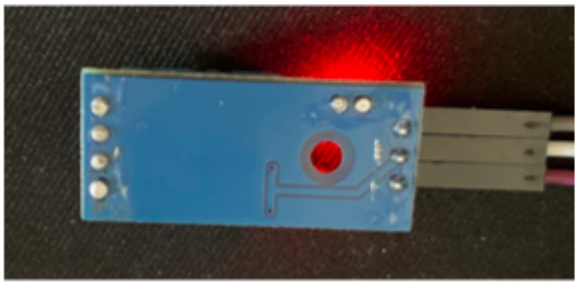
   “He de decir que inicié este tipo de pruebas con la placa Arduino Uno R3, ya que la información que encontraba era con esta placa, cosa que luego tuve que adaptar a Nodemcu ESP8266”.

2. Bien. Las partes necesarias son:
   a. Placa Arduino Uno R3: se puede conseguir en los establecimientos que hemos comentado al inicio de este documento (Alibaba y Amazon)

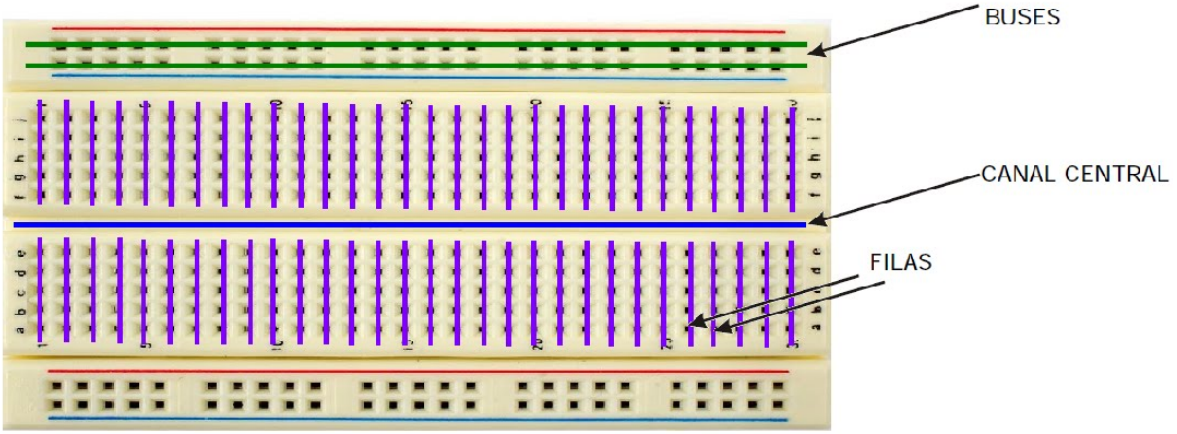


b. Sensor DHT11.

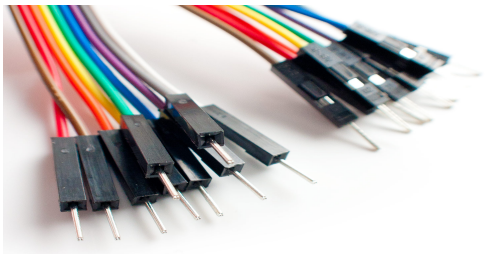




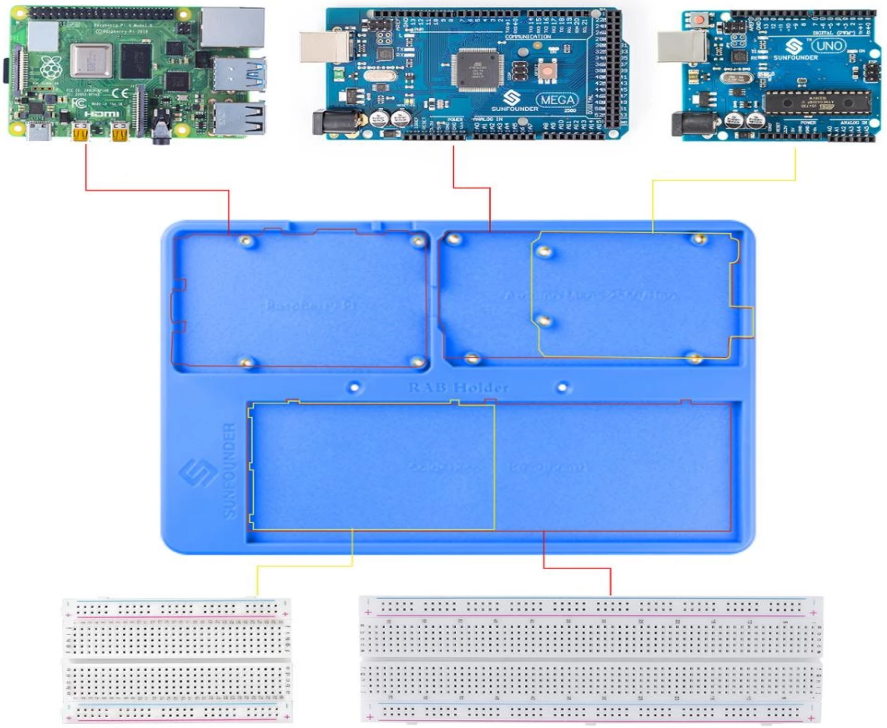
c. Placa Protoboard: es un tablero con orificios, conectados entre sí de manera interna y lineal, que nos permiten insertar componentes electrónicos, cables y demás.



d. Cableado de diferentes tipos de pines, ya que depende del tipo de sensor, necesita un tipo de pin u otro.



e. Y si es necesario, una placa de soporte.



3. Ahora el programa. Se puede hacer de varias maneras: Una sería hacerte tú mismo el programa, previos conocimientos, claro está. Y dos, usas uno “prefabricado” y adaptarlo. Admito que usé uno prefabricado, dado que mis conocimientos eran nulos, no tenía ni idea de por dónde empezar. (Me gustaría que, al finalizar el proyecto, poder afirmar que he adquirido más conocimientos sobre este tipo de programas).

Programa Placa Nodemcu ESP8266 V3 y Sensor DHT11:

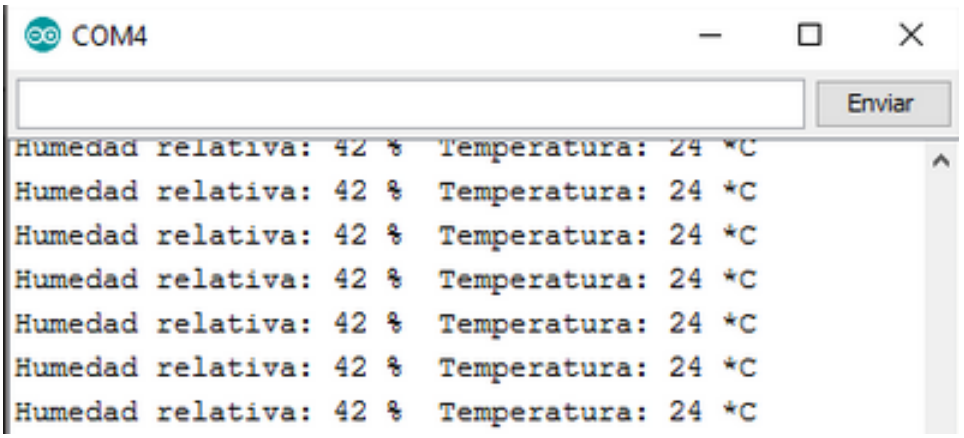
```
#incluye "DHT.h" // incluye la biblioteca del sensor de temperatura y humedad DHT11
#define DHTTYPE DHT11 // DHT 11
#define dht_dpin D6
DHT ( dht_dpin, DHTTYPE ) ;
void setup ( void )
{
        dht.begin () ;
        Serial.begin ( 9600 ) ;
        delay ( 700 ) ;

}
void loop () {
        float h = dht.readHumidity () ;
        float t = dht.readTemperature () ;
        Serial.print ( "Humedad relativa:  ") ;
        Serial.print ( h ) ;
        Serial.print ( “%\n" ) ;
        Serial.print ( "Temperatura:  ") ;
        Serial.print ( t ) ;
        Serial.println ( "*C" ) ;
    delay ( 800 ) ;
}
```

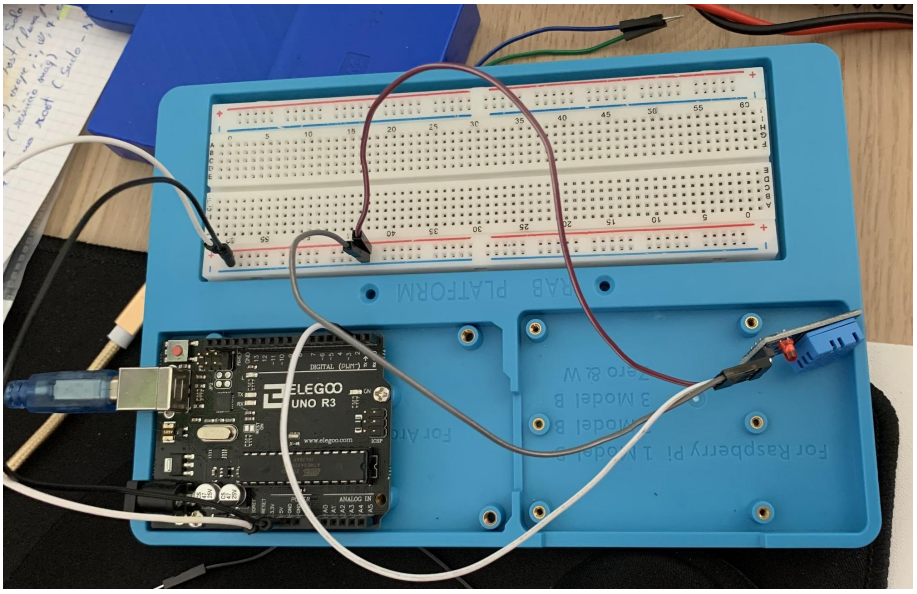
4. Subimos el programa, como lo explico en el punto:

Prueba fun… con placa ESP8266 V3 con IDE Arduino. Punto 4º.

5. Ahora abrimos el monitor serie de Arduino:
   Herramientas > monitor série, se abre una ventana pequeña mostrando los datos de los sensores.

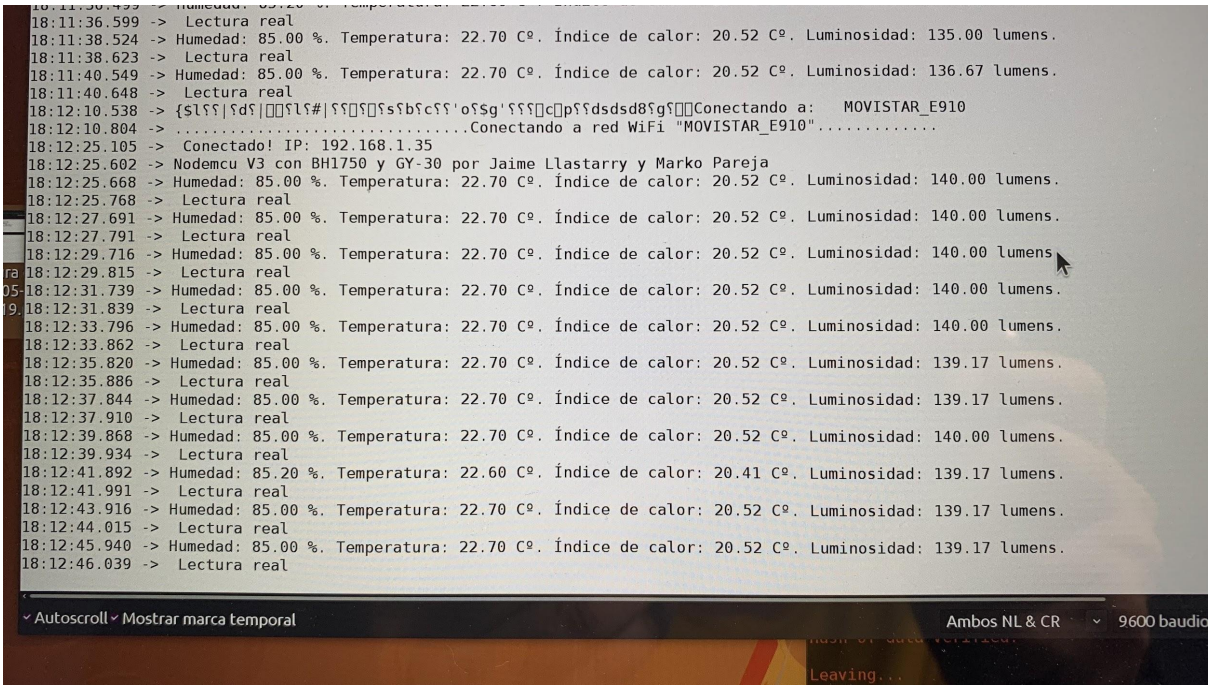


6. Y vemos el montaje de este proyecto.



Pruebas con el medidor: Fusión de sensores.

En este punto no me voy a extender mucho ya que lo único que he hecho ha sido buscar, e ir insertando los datos de programación en un mismo programa. Ésto se ha hecho para unificar la lectura de varios sensores diferentes, en un solo programa y así, obtener los datos deseados de una sola vez.



Me he visto obligado a prescindir de otro sensor que utilizaba, el BMP180, ya que era incompatible con el DHT11. Lo único diferente es que no muestra el índice de calor en las lecturas. Por lo demás, el proyecto sigue avanzando y próximamente, se concluirá.

Rectifico. Mediante otra biblioteca, que he encontrado, del programa del sensor DHT11, he conseguido que muestre el índice de calor.

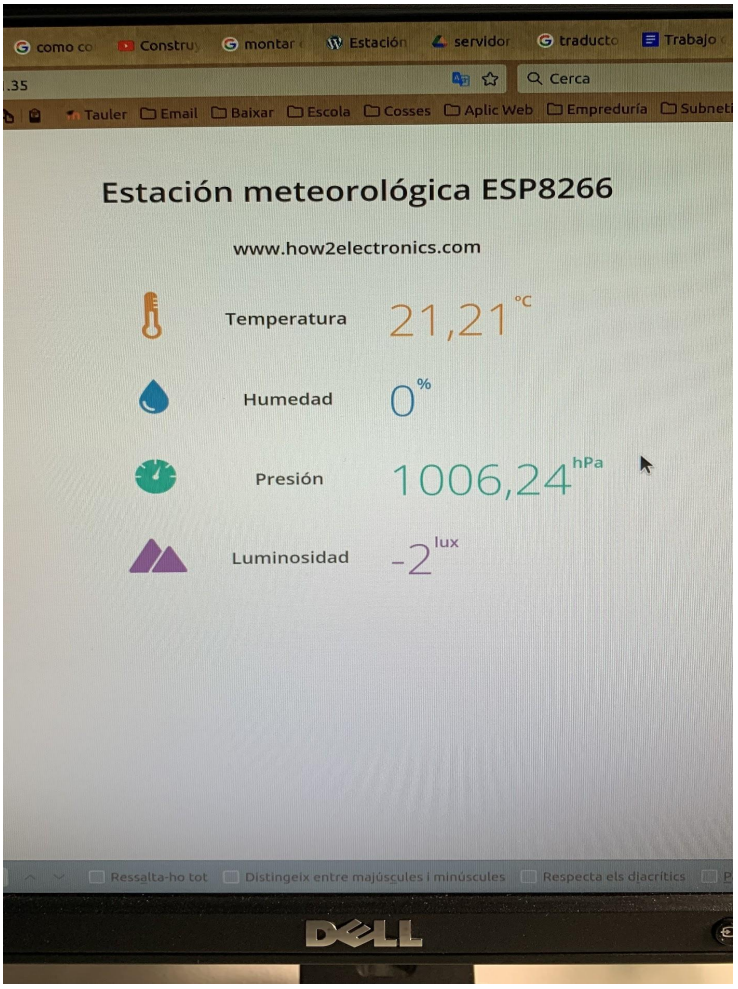
Pruebas con el medidor: Mandar datos a la web.

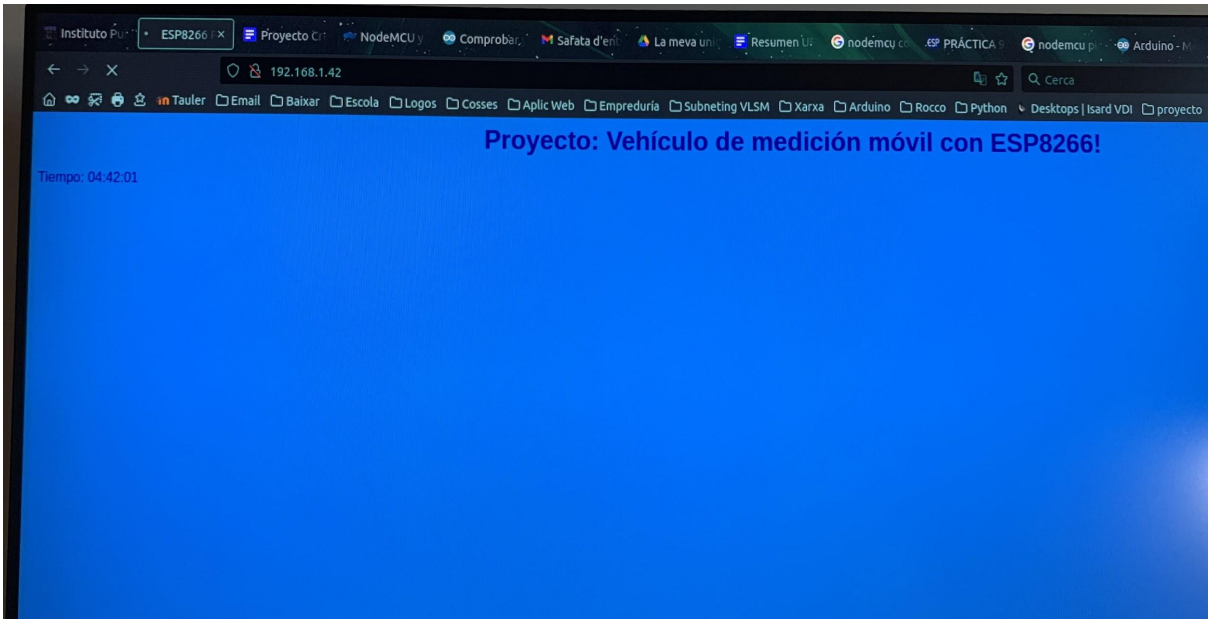
A partir de ahora toca sufrir. Este es el punto final del proyecto del medidor. Acto seguido fusionamos las dos partes y veremos que tal va todo. Pero antes acabemos este.

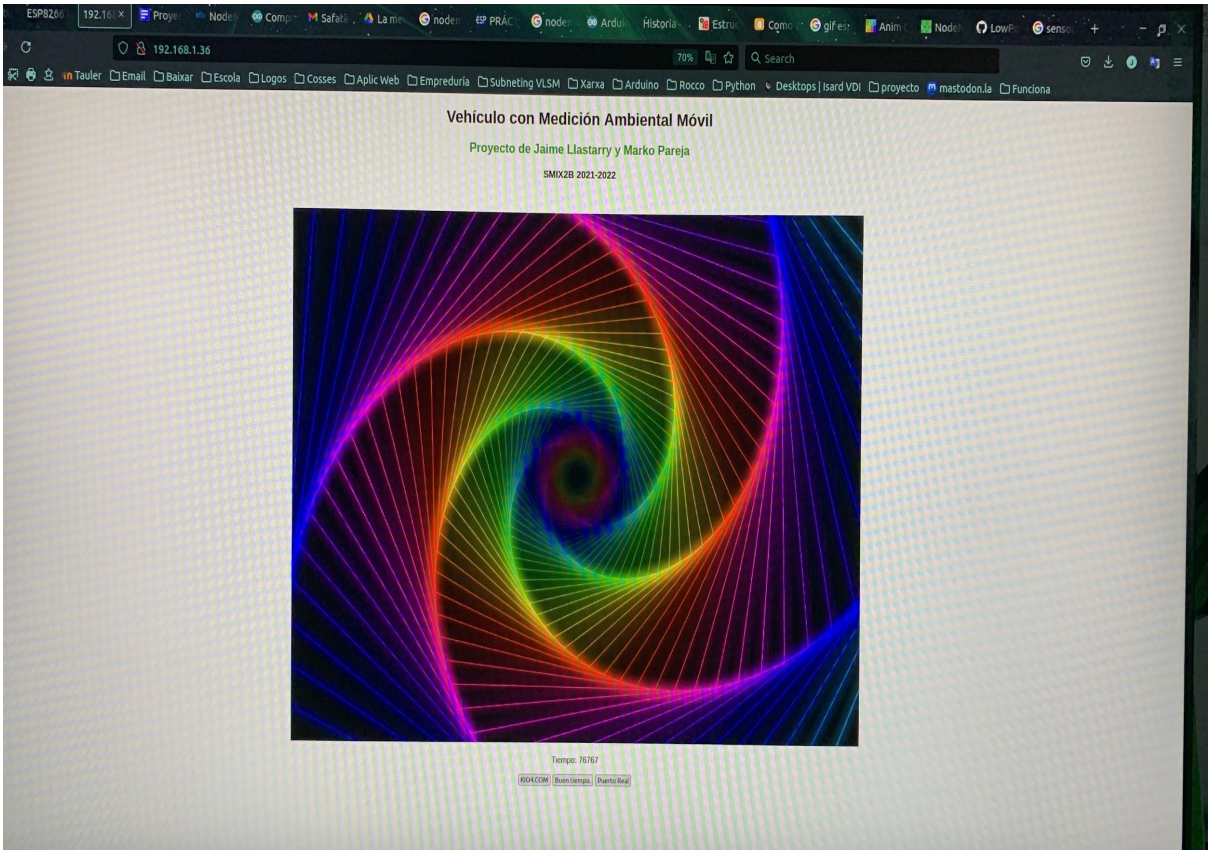
¡¡Esta parte, para mí, es la más costosa y metódica que he estado haciendo.!!

Me explico: Al obtener los datos de los sensores y ver que funcionan correctamente, llega el momento de que muestre esta información en una página web. Uff!!. Los datos que he obtenido, a día de hoy, son muy liosos, puesto que se pueden utilizar varias bibliotecas y programas para ello, pero he de encontrar la que se adapte mejor a mis necesidades y no es fácil, Ya he probado unas 3, y no veo los resultados esperados.

Permítanme que les muestre unas imágenes.







Estas imágenes superiores, son el resultado de ir probando varios tipos de programas con los datos que yo quería incluir. Éstos no son el resultado que quería para este proyecto.

Al fin, lo he conseguido, ahora ya llegan las mediciones hasta una página web. Me ha complicado mucho este punto, ya que no esperaba encontrarme con tantas, y diferentes, informaciones sobre lo mismo. No he encontrado dos programas que contengan el mismo tipo de información sobre cómo mandar los datos a la web. Sobre éste modelo, he adaptado mi programa con el siguiente resultado (imagen superior). He de decir que es una página web muy básica, pero para mí, lo más importante es que muestre el resultado de las mediciones. Claro que podía poner imágenes de fondo, iconos y otras cosas, pero, como he dicho antes, este punto me ha llevado más tiempo del que yo esperaba.

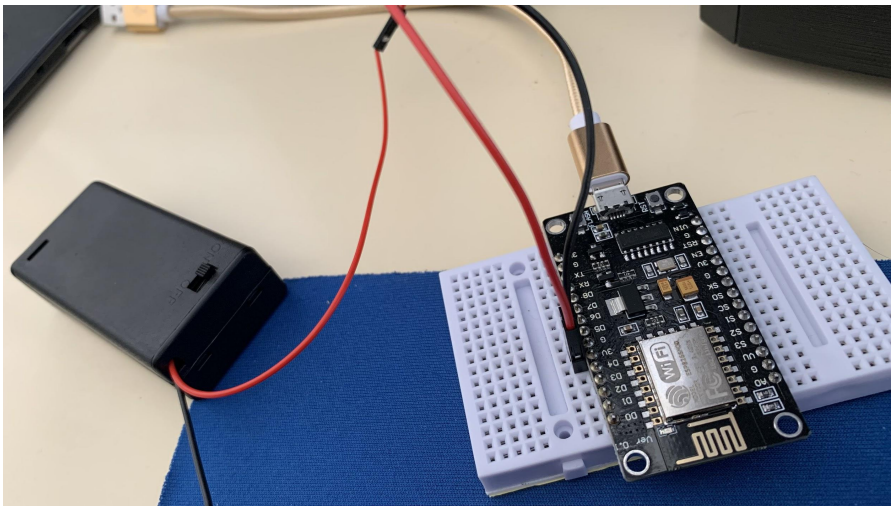
Prueba final medidor: Alimentar los componentes.

Hasta aquí ... todo bién, pese a las dificultades superadas. Ahora es el momento definitivo, y yo me planteo, ¿cómo alimentar los componentes del proyecto, ya que es una estación (mini) meteorológica móvil?

Para que toda esta parte funcione, he de alimentar eléctricamente todas sus partes correctamente, sin subir demasiado el voltaje, ya que si nos sobrepasamos podemos quemar algún componente. Y si por el contrario, no ponemos el suficiente voltaje, no funcionará nada.

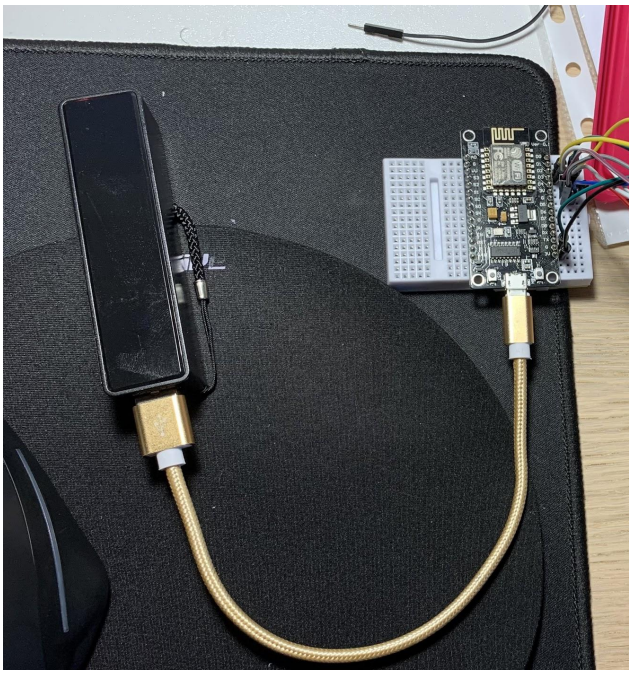
Cómo no he tenido muy presente este punto en un principio, no me he preocupado demasiado, ya que mientras se programa y se prueba todo, estamos conectados al ordenador con el cable tipo Macho USB - Macho Micro Usb, y no piensas en la parte de conexionado eléctrico portátil.

En un principio, se utilizan dos pilas tipo AA-LR6 (3V en total), conectándolas a los pines 3V y G de la placa. Funciona bien.



¡¡MUY IMPORTANTE!! Si se usan tres pilas, 4,5V, se quema la placa ESP8266!!

La duración de las pilas es escaso (entre 4 y 6 horas), en las pilas de usar y tirar (no he probado con las recargables). Por casa tenía una Power-Bank que justo dá la potencia del puerto usb del ordenador, 5V y 1A. La he utilizado varias veces con este proyecto y tiene una duración de alrededor de 18 horas, esta duración es un poco corta, ya que este dispositivo es un poco antiguo. Cuyo resultado es el esperado: todo funciona sin cables conectados al ordenador.



5- Ponerlos a prueba conjuntamente.

Ha llegado la hora de probar las dos partes en una sola. Estábamos un poco nerviosos por ver cómo nos podía quedar todo en conjunto: unir dos ideas en una sola, y que encima, funcione, no lo teníamos muy claro. Decidimos centrarnos cada uno en su parte e ir contándonos los progresos que íbamos haciendo, al igual que los fracasos y, de igual manera, teníamos que montar y desmontar ambas partes, para ver todo en conjunto.

La parte del transporte, que a priori, era la más “fácil”, resultó no serlo tanto, al igual que la parte del medidor, con sus variaciones de códigos y calibrado de los sensores, se tardó más de lo esperado.

Llega el momento de probarlo todo varias veces en varios entornos; nuestros domicilios, en el aula del centro y en un garaje de mi propiedad.

Los problemas a la hora de unirlos no fue casi ninguno, en todos los espacios parece funcionar; el medidor envía la señal con los datos a bastante distancia, para el coche no le resulta ninguna dificultad llevar más peso (aquí hemos conseguido añadir sólo un peso de 180 gramos), y para el medidor le va genial estar tapado, menos los sensores que, obviamente, tienen que tener contacto con el exterior. Decimos “casi” porqué si no conseguimos señal wifi, toda la parte del Medidor no funciona. Se ha pensado en compartir los datos y la señal wifi desde un Iphone y, sinceramente, va como anillo al dedo; cero fallos, cero desconexiones.

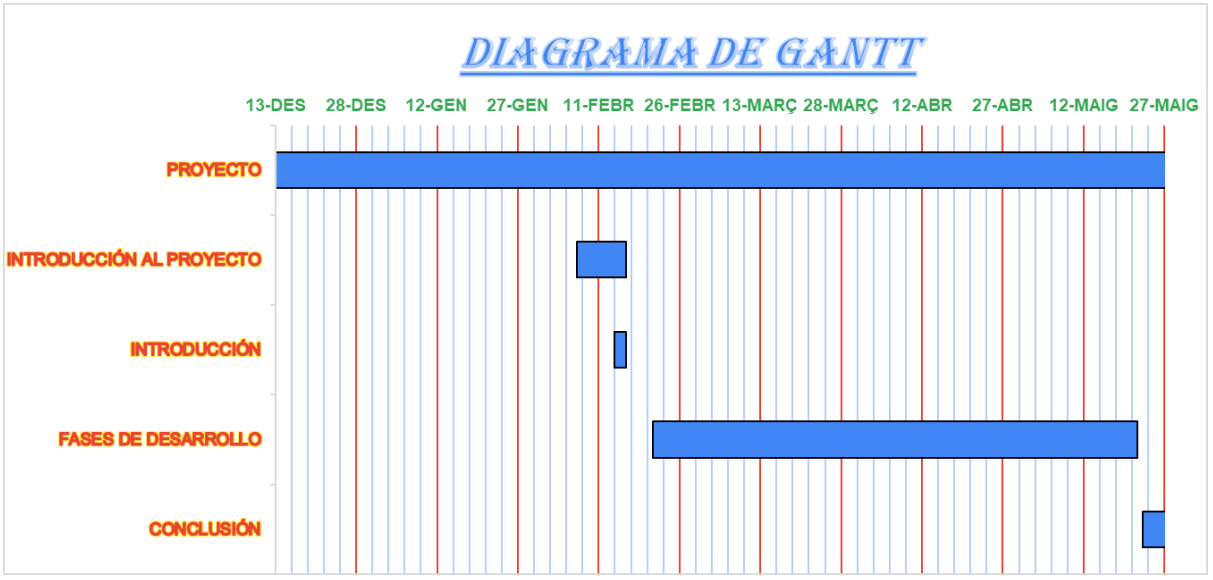
4 Conclusión y Diagrama de Gantt.

Creemos que con este proyecto lograremos el objetivo que buscamos:

- ➢ Obtener datos en diferentes zonas para su comparación y uso de los datos ambientales obtenidos, y llegar a la conclusión de dónde hace mejor tiempo (en tiempo real), para poder adaptarnos al entorno, dependiendo de nuestras necesidades.

- ➢ Si se hubiera pensado en un vehículo con otras características, como pueden ser; unas ruedas para ir por la tierra, una estructura para la medición completamente hermética, sensores y acabados de mejor calidad, más variedad de sensores y más tiempo para su preparación, tendríamos entre manos un vehículo que, sin importar la ubicación en la que se encontrase, diera las mediciones como una estación meteorológica fija. Incluso se podría usar en catástrofes para obtener otros datos: calidad del aire, fugas de Gas, incendios, etc.

- ➢ Como se puede observar en el siguiente diagrama, detectamos que se tarda más tiempo en desarrollar el proyecto (que es lo normal), que en explicar e intentar desarrollar todo en detalle: programas y demás partes del proyecto.



5 Webgrafia.

Compras:

- https://www.amazon.es/
- https://es.aliexpress.com/
- Otros:
    - https://hipermontigala.com/

Información:

- Nodemcu:
    - Anexo 7
    - https://www.nodemcu.com/index_en.html
- Firmware:
    - Anexo 7
    - https://archive.org/details/sim_datamation_1967-01_13_1/page/22/mode/2up
    - https://archive.org/details/pub_datamation
- Github:
    - Anexo 7
    - https://es.wikipedia.org/wiki/GitHub
- Lua:
    - Anexo 7
    - https://es.wikipedia.org/wiki/Lua
- Imagen:
    - Dibujo Nodemcu :
    - https://electronilab.co/tienda/nodemcu-board-de-desarrollo-con-esp8266-wifi-y-lua/
    - Logo Nodemcu:
    - https://www.nodemcu.com/index_en.html
    - https://diyi0t.com/what-is-the-esp8266-pinout-for-different-boards/
    - Placas Nodemcu:
    - https://diyi0t.com/what-is-the-esp8266-pinout-for-different-boards/
- Arduino
    - https://www.arduino.cc/en/software
- Wokwi
    - https://wokwi.com/projects/new/arduino-uno?lang=es-ES

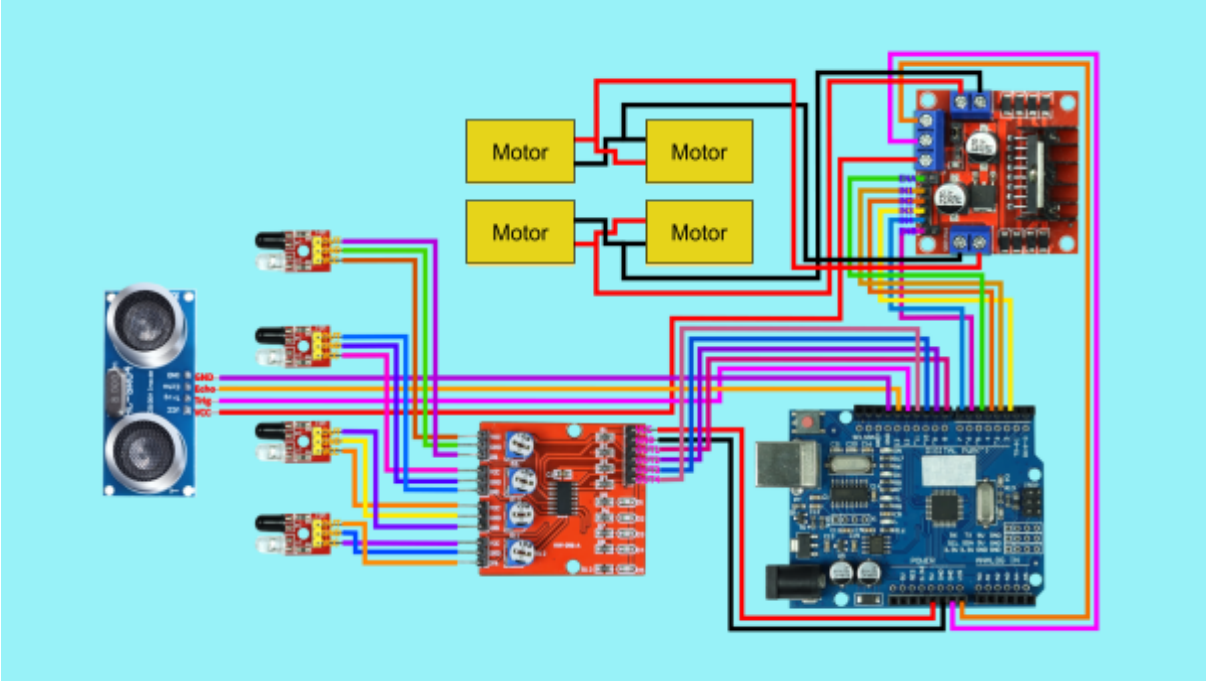
- Estructura programa
    - https://openlanuza.com/estructura-de-un-programa-en-arduino/

- Guia página web Medidor
    - https://lastminuteengineers.com/esp8266-dht11-dht22-web-server-tutorial/#google_vignette

- Creative Commons
    - https://creativecommons.org/licenses/by-nc/4.0/deed.es

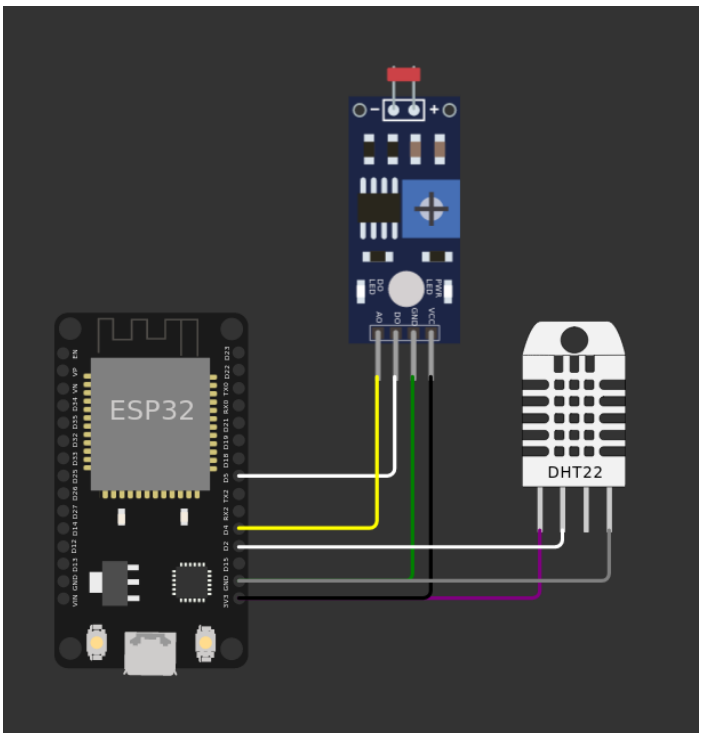
6 Anexo: Cableado.

Robot:



Medidor:

La imagen varía de la realidad en los elementos que la forman.



Sensor DHT11:

- VCC = Pin Corriente a 3 V. (lila)
- GND = Pin Tierra (gris)
- DATOS = Pin D4 (blanco)

Sensor GY-30

- VCC = Pin Corriente a 3 V. (negro)
- SCL = Pin D3 (blanco)
- SDA = Pin D2 (amarillo)
- AOO = Vacío
- GND = Pin Tierra (verde)

Para más detalles, consultar en el apartado **9 Manual de usuario.**

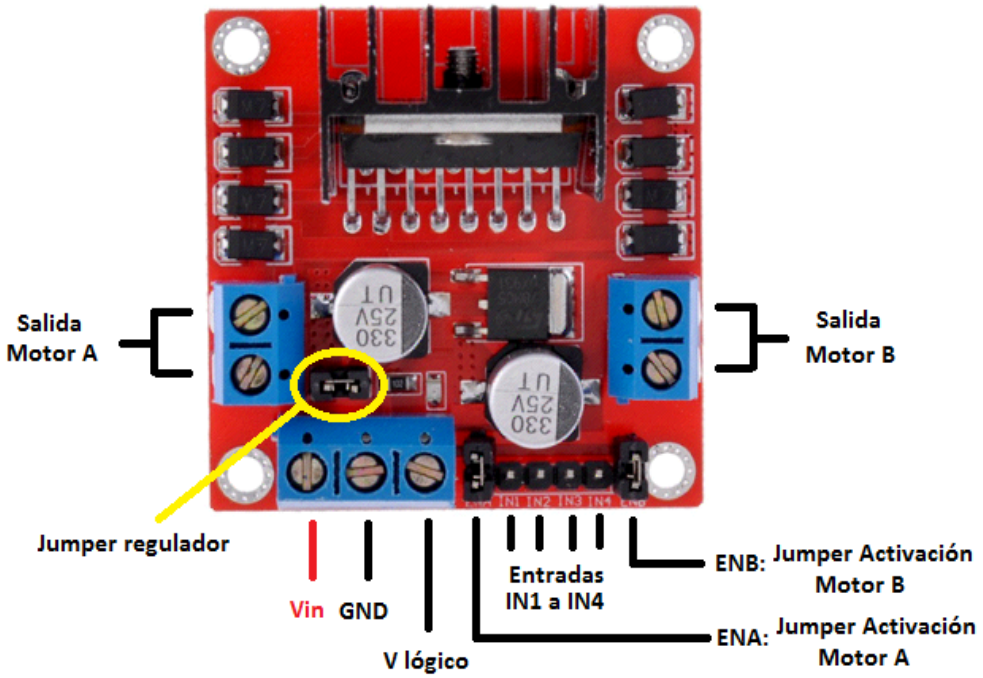
7 Anexo 1: Información e Instalación de programas.

Robot:

Información sobre hardware y software

L298N

En este montaje tambien tenemos a mas a mas de lo mencionado en puntos anteriores, un servo motor en nuestro caso el L298N, el módulo controlador de motores L298N H-bridge nos permite controlar la velocidad y la dirección de dos motores de corriente continua o un motor paso a paso de una forma muy sencilla, gracias a los dos H-bridge que monta, básicamente un puente-H o H-bridge es un componente formado por 4 transistores que nos permite invertir el sentido de la corriente, y de esta forma podemos invertir el sentido de giro del motor.
El rango de tensiones en el que trabaja este módulo va desde 3V hasta 35V, y una intensidad de hasta 2A. A la hora de alimentarlo hay que tener en cuenta que la electrónica del módulo consume unos 3V, así que los motores reciben 3V menos que la tensión con la que alimentamos el módulo.
Además el L298N incluye un regulador de tensión que nos permite obtener del módulo una tensión de 5V, perfecta para alimentar nuestro Arduino. Eso sí, este regulador sólo funciona si alimentamos el módulo con una tensión máxima de 12V.
Es un módulo que se utiliza mucho en proyectos de robótica, por su facilidad de uso y su reducido precio.



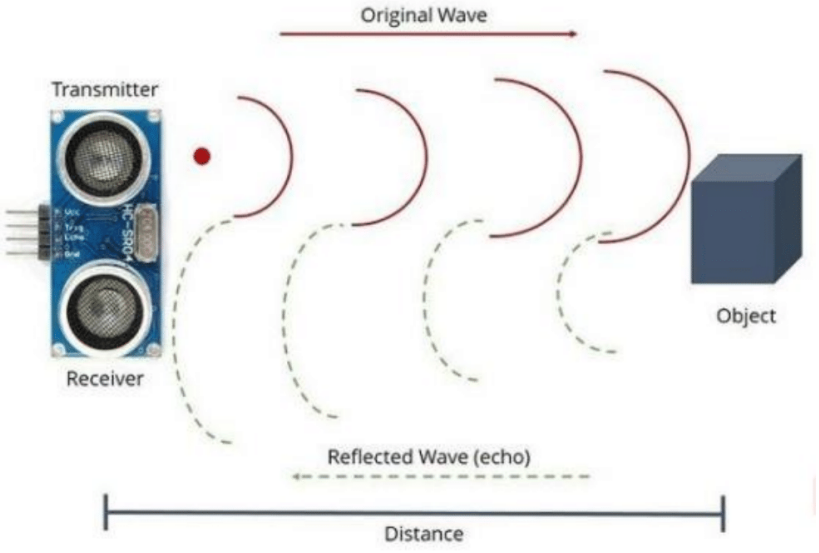
Este módulo nos deja una tabla como chuleta para hacer funcionar los motores esta es :

State	ENA	ENB	IN1	IN2	IN3	IN4
Stop	0	0	X	X	X	X
Brake	1	1	0	0	0	0
Forward	1	1	1	0	1	0
Back	1	1	0	1	0	1
Turn Left	1	1	0	0	1	0
Turn Right	1	1	1	0	0	0

Este esquema representa donde tenemos que enviar corriente para que vaya de una manera o otra, si enviamos corriente a los pines A B y 3 nuestro coche iría hacia la izquierda pero por ejemplo si enviamos corriente a los pines A B 1 y 3 el coche iría hacia delante, este esquema sirve mucho para hacer el código.

HC-SR04

Sobre este módulo ya hemos hablado pero ahora vamos a explicar como trabaja, este módulo calcula las distancias mediante ultrasonido es decir emite una onda de sonido y según el rebote que reciba es una distancia o otra, el esquema sería tal que así :



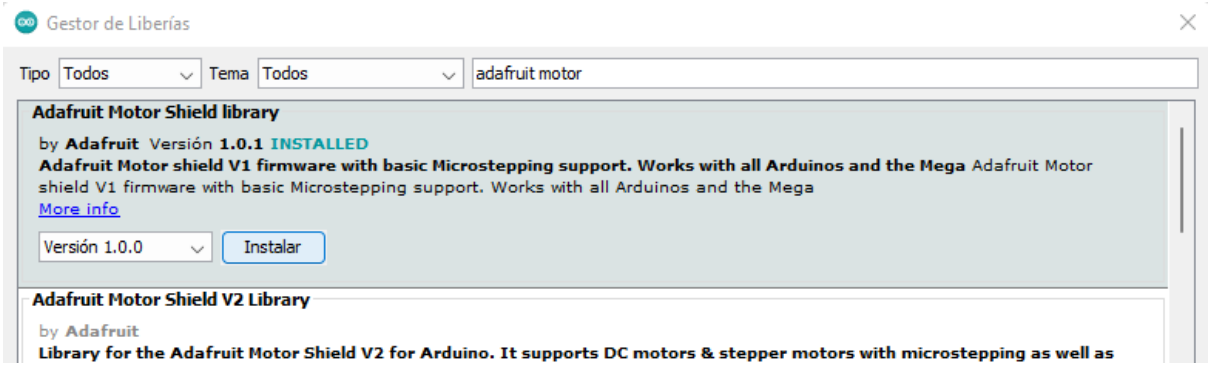
Gracias a este módulo podemos incluir en el código variables con las que determinar que si la distancia es menor que 10 cm por ejemplo enviar corriente a una serie de pines vista anteriormente y así que de marcha atrás hasta que haya más de 20 cm por ejemplo, esta función ha sido la que he implementado en el código.

Instalación arduino

Primero entraremos en la web oficial de descarga Arduino https://www.arduino.cc/en/software una vez dentro procederemos a seleccionar nuestro sistema operativo en mi caso Windows tenemos tres opciones, instalarlo en un .msi que es un microsoft software installer, la otra opción es en un .zip para descomprimirlo y ejecutarlo directamente o la que elegí yo que es mediante la Microsoft Store es más sencillo como si fuese un app.

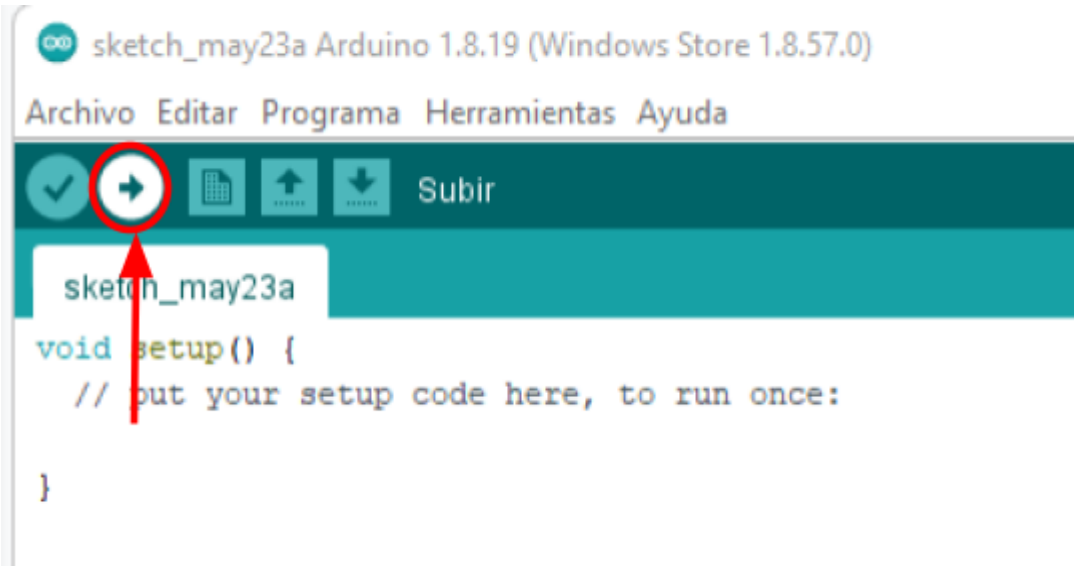
Puesta a punto de la app Arduino

Primero tenemos que instalar librería que nos reconocerán los componentes de nuestro proyecto, dentro de la app le daremos a “Programa > Incluir Libreria > Administrar Bibliotecas” una vez dentro tenemos que instalar Adafruit Motor Shield Library, para encontrarlo buscaremos Adafruit motor y nos saldrá el primero, seleccionamos la versión más reciente y le daremos a instalar.



Una vez hecho esto tendremos que ir a “Herramientas > Puerto > (Nuestro puerto USB donde esta conectada la placa Arduino)”.

Una vez hecho esto solo tendremos que programar nuestro código e ir probando dándole a “Subir”, nos puede dar error si tenemos algún fallo en la sintaxis o algo parecido pero si no da error ya estará subido a la placa.



Medidor:

Placa Nodemcu y sus versiones.

El firmware NodeMCU fue creado poco después de aparecer el ESP8266, en diciembre del 2013. Poco después, en octubre del 2014, se hizo pública la primera versión del firmware NodeMCU en GitHub. Más tarde se publicaba la primera placa NodeMCU, denominada devkit v0.9, siendo Open Hardware.

En esos primeros momentos del ESP8266, la información era poca y confusa. El interés de la comunidad se limitaba al ESP01, que se consideraba un módulo Wifi barato para procesadores como Arduino.

El firmware NodeMCU podía grabarse en un ESP8266, es decir, se podía programar con el lenguaje script Lua. La programación el Lua permitía la conexión y programación del ESP8266 de una forma mucho más sencilla que las herramientas oficiales proporcionadas por Espressif.

"Lua es un lenguaje de programación imperativo y estructurado, creado en 1993 por Roberto Ierusalimschy, Luiz Henrique de Figueiredo y Waldemar Celes. Está basado en C y Perl, y está diseñado para ser muy ligero. Ha sido implementado en una gran variedad de dispositivos, desde videoconsolas a robots industriales. La página oficial es https://www.lua.org/."

Con el paso del tiempo y la aparición de otras alternativas para programar ESP8266, como C++ con el entorno del Arduino y otras como Micro Python, el interés en Lua ha disminuido considerablemente.

A pesar de que la programación en Lua tenía aspectos interesantes, no es un lenguaje tan extendido como C++ y Python. Además, nunca consiguieron hacerlo totalmente estable en el ESP8266. Por otro lado, al ser un lenguaje interpretado (en lugar de compilado) el rendimiento y aprovechamiento de los recursos es inferior.

En 2015 el equipo de desarrollo original dejó de mantener el firmware de NodeMCU. Aunque sigue siendo mantenido por una comunidad de desarrolladores el interés en el firmware ha caído casi por completo. Por este motivo, actualmente nos referimos con NodeMCU más a la placa de desarrollo que al firmware.

No obstante, aunque actualmente el firmware haya caído un poco en el olvido, no hay que olvidar la contribución que el proyecto NodeMCU, tanto firmware como placa de desarrollo, han supuesto para la proliferación e implantación del ESP826.

- Versiones.

Vamos a ver tres versiones de la placa esp8266. Hay alrededor de 14, pero me centraré en los modelos de la imagen superior. Hago un pequeño resumen para ver las diferencias entre ellas:

1. ESP01:

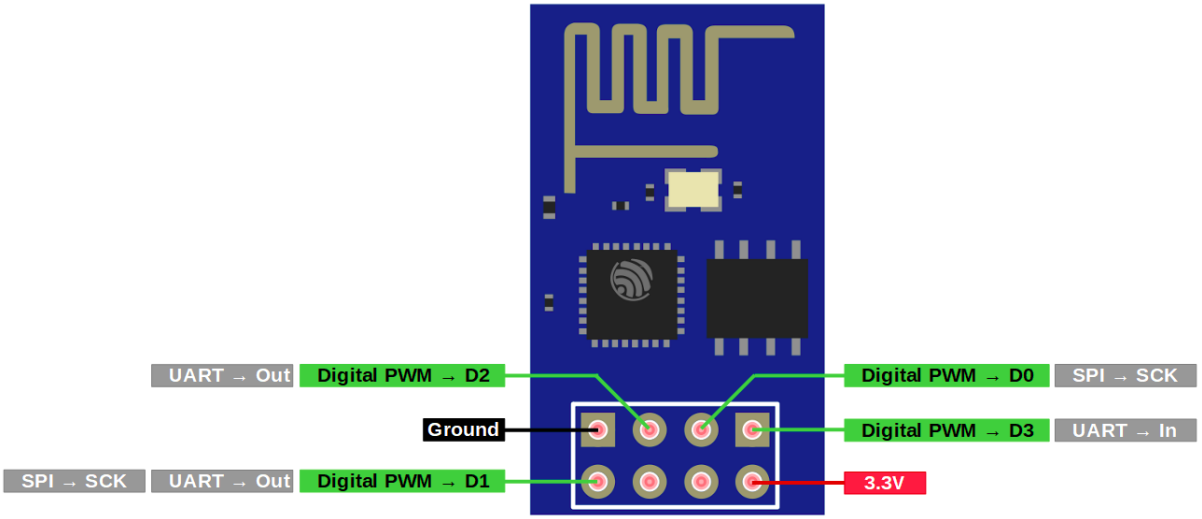
    a. El ESP-01 es un módulo pequeño basado en el microcontrolador ESP8266.

    b. Este módulo mide 25 mm de alto por 14 mm de ancho y tiene 8 GPIO o pines.

    c. Como todos los módulos basados en el ESP8266 tiene WiFi incluido.

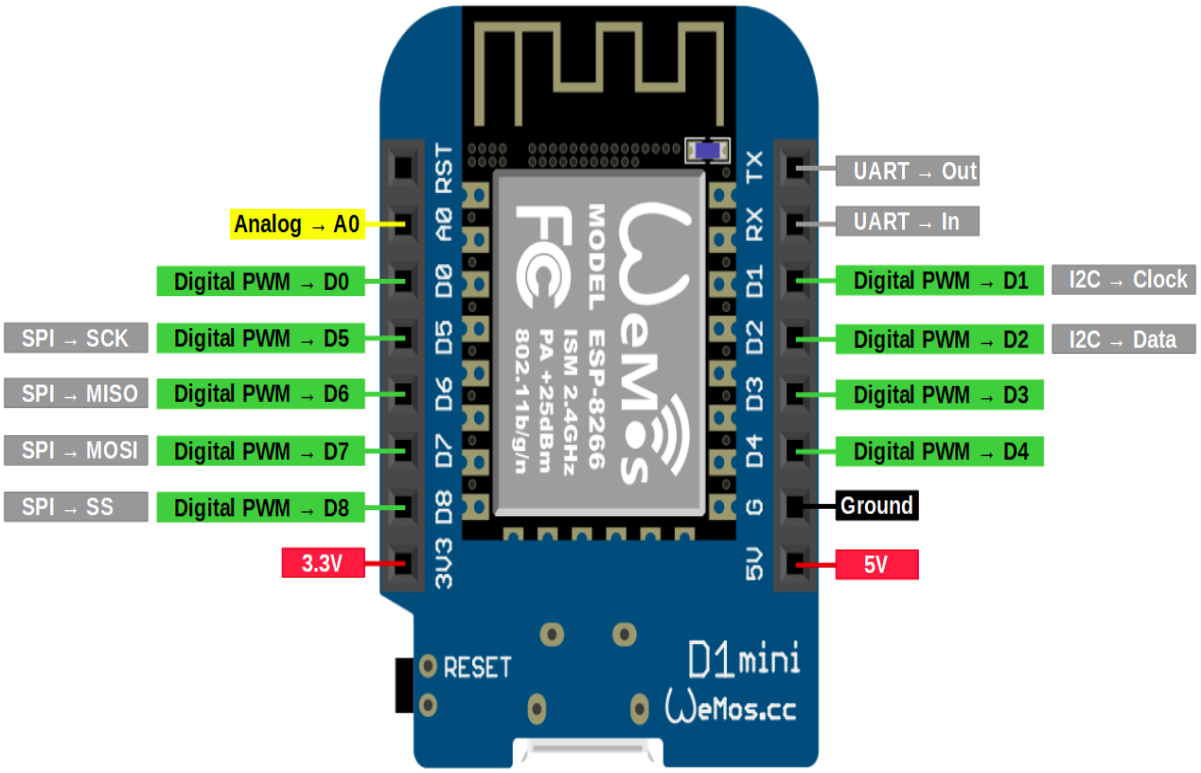
    d. Hay una versión flash de 1 MB y una versión flash de 512kB en el mercado.

    e. El ESP-01 no puede conectarse mediante USB. La única posibilidad es conectarse a una fuente de alimentación de 3,3 V a través del pin VCC.



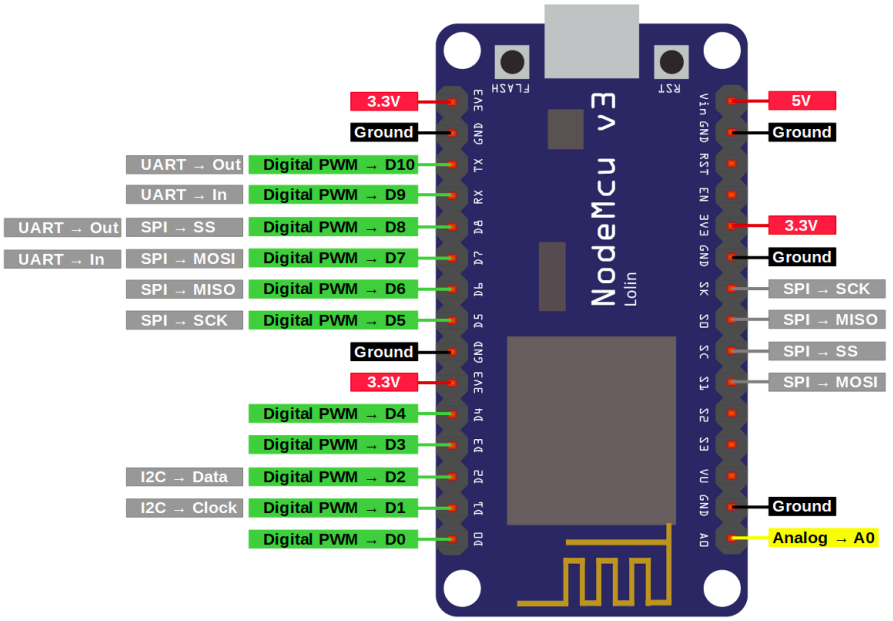
2. WeMos D1 Mini:

    a. Su tamaño de 34,2 mm de alto x 25,6 mm de ancho se encuentra entre el módulo NodeMCU y el ESP-01.
    b. Existen diferentes versiones de la placa WeMos D1 Mini.
    c. El WeMos D1 Mini, con ESP8266 integrado, tiene un voltaje de funcionamiento y de salida, de 3,3V y 5V para alimentar diferentes componentes externos, y una memoria flash de 4MB.
    d. Los dos pines, 3,3V y 5V, pueden proporcionar hasta 500 mA. Además, el pin de 5V puede alimentar toda la placa con un voltaje de suministro entre 4V y 6V.
    e. El circuito está cerrado sobre un pin de tierra.



3. NodeMcu:

   a. La placa de NodeMCU obtuvo su nombre de una plataforma IoT de código abierto. Comercialmente se llama NodeMCU o ESP8266E-12.
   b. La plataforma incluye firmware que se ejecuta en el SoC WiFi ESP8266 de Expressif Systems y el hardware que se basa en el módulo ESP-12. El NodeMCU incorpora una memoria flash de 4 MB.
   c. Hay diferentes versiones de la placa:
      i. La versión 1 es bastante antigua y no recomendaría usarla para proyectos futuros.
      ii. El V2 corrige las deficiencias de la placa V1, y encaja muy bien en las placas base. El chip se actualizó de un ESP-12 a un ESP-12E.
      iii. La versión 3 es inventada por el productor LoLin con mejoras ligeramente menores. Cuenta con un puerto USB más robusto. LoLin decidió usar uno de los dos pines de reserva para la salida de USB y el otro para una conexión a tierra adicional. Pero el mayor inconveniente de esta placa es su tamaño. Las dimensiones del V2 son 48x26x13 mm en comparación con el V3 con 58x31x13 mm. La placa V3 no cabe en una placa de pruebas pequeña.



Antes he hablado de firmware, pero ¿que és?. (Viene de la pág. 22).

Se conoce como firmware al conjunto de instrucciones de un programa informático que se guarda en una memoria ROM, flash o similar. Estas instrucciones fijan la lógica primaria que ejerce el control de los aparatos. Éste término parece ser que tiene su origen en la década de los años 60. Afirmamos que fué en el año 1967, cuando salió publicado en la revista Datamotion, un artículo de Rudy Meléndez (adjunto más abajo "Más información", imagen del artículo original y enlace web, ya que los datos que hay en España, no están muy bien. Lo he encontrado en EEUU, explicando que era el firmware). Dicho nombre, el firmware, hace referencia a la programación en firme, formando parte del hardware ya que se encuentra integrado en la electrónica, pero también está considerado como parte del software al estar desarrollado bajo lenguaje de programación. Es decir, que el firmware es la unión entre las instrucciones que llegan al dispositivo desde el exterior (programación) y sus diversas partes electrónicas. Más concretamente, podemos establecer que el firmware de cualquier dispositivo, lo que hace es cumplir tres claras funciones:

- 1ª: Logra otorgar al sistema las rutinas básicas de funcionamiento y respuesta con respecto a las peticiones que recibe.
- 2ª: Otra de las funciones, es establecer una sencilla interfaz para que se pueda introducir, fácilmente, la configuración del sistema mediante una serie de parámetros.
- 3ª: Y otra de las funciones que posee todo firmware, es controlar y gestionar tanto lo que es el arranque del dispositivo, como el mismo arranque del sistema.

Los microprocesadores, las impresoras, los monitores y los chips de memoria son algunos de los dispositivos que cuentan con firmware. Un ejemplo de firmware es el programa BIOS de la computadora, que comienza a funcionar apenas se enciende la máquina.

Existen tres tipos de BIOS, que se clasifican en base al método que se ha empleado para grabarlos:

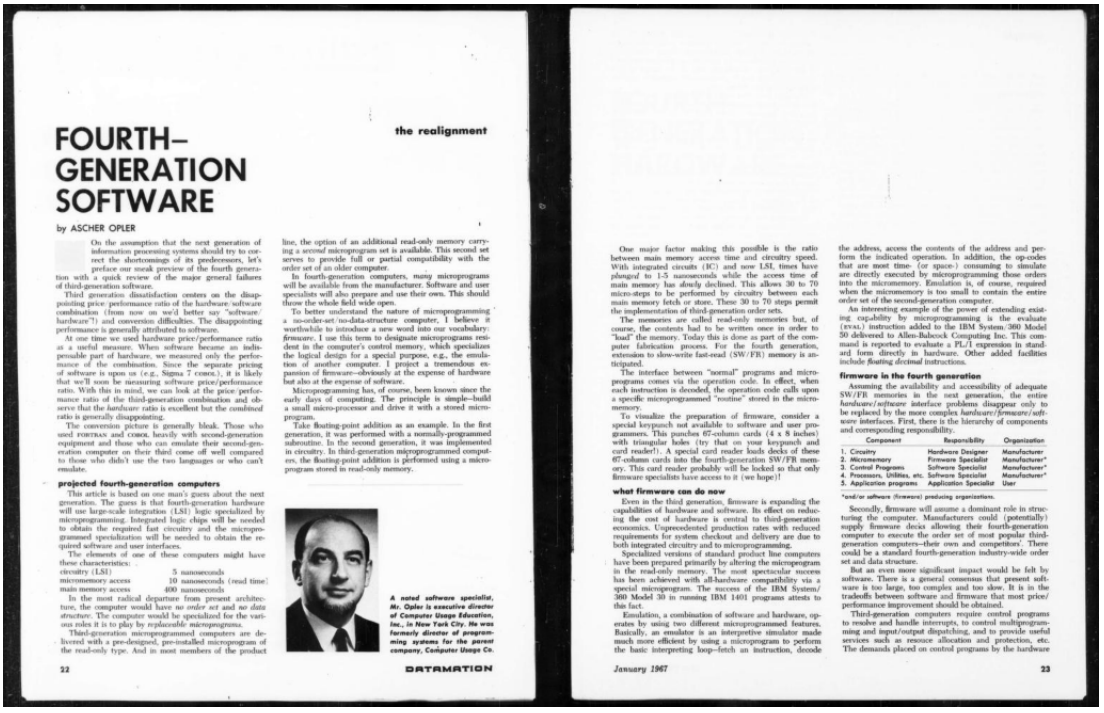
- ROM. Se graba en el momento en el que se crea el chip y su información ya no se puede modificar.
- PROM. Funciona de manera similar a las memorias de clase ROM pero sólo se puede escribir una única vez.
- EPROM. Funciona como las citadas ROM pero tiene la novedad de que permite borrarse y escribirse tantas veces como sea conveniente.

La arquitectura de una computadora está formada por distintas capas con diferentes niveles de abstracción. El hardware es la base y luego aparece el firmware. Sobre él se acumulan el ensamblador, el kernel, el sistema operativo y, al final, las aplicaciones.

Cabe destacar que el usuario, cuenta con la posibilidad de actualizar el firmware para solucionar errores o añadir prestaciones. Estas actualizaciones son problemáticas, ya que si se produce algún fallo en la actualización, el dispositivo puede dejar de funcionar.

Dicho de otra manera, podría decirse que es la unificación entre órdenes externas que van al dispositivo y la electrónica del mismo.

(Imágenes del artículo publicado en los años 60, el original).
(Más abajo, añado el enlace directo al artículo).



FOURTH–GENERATION SOFTWARE

by ASCHER OPLER

the realignment

On the assumption that the next generation of information processing systems should try to correct the shortcomings of its predecessors, let's preface our sneak preview of the fourth generation with a quick review of the major general failures of third-generation software.

Third generation dissatisfaction centers on the disappointing price–performance ratio of the hardware/software combination (from now on we'd better say "software/hardware"!) and conversion difficulties. The disappointing performance is generally attributed to software.

At one time we used hardware price/performance ratio as a useful measure. When software became an indispensable part of hardware, we measured only the performance of the combination. Since the separate pricing of software is upon us (e.g., Sigma 7 comes.), it is likely that we'll soon be measuring software price/performance ratio. With this in mind, we can look at the price/performance ratio of the third-generation combination and observe that the *hardware* ratio is excellent but the combined ratio is generally disappointing.

The conversion picture is generally bleak. Those who used FORTRAN and COBOL heavily with second-generation equipment and those who can emulate their second-generation computer on their third come off well compared to those who didn't use the two languages or who can't emulate.

projected fourth-generation computers

This article is based on one man's guess about the next generation. The guess is that fourth-generation hardware will use large-scale integration (LSI) logic specialized by microprogramming. Integrated logic chips will be needed to obtain the required fast circuitry and the microprogrammed specialization will be needed to obtain the required software and user interfaces.

The elements of one of these computers might have these characteristics:

circuitry (LSI)	5 nanoseconds
micromemory access	10 nanoseconds (read time)
main memory access	400 nanoseconds

In the most radical departure from present architecture, the computer would have *no order set* and *no data structure*. The computer would be specialized for the various roles it is to play by *replaceable microprograms*.

Third-generation microprogrammed computers are delivered with a pre-designed, pre-installed microprogram of the read-only type. And in most members of the product line, the option of an additional read-only memory carrying a *second* microprogram set is available. This second set serves to provide full or partial compatibility with the order set of an older computer.

In fourth-generation computers, many microprograms will be available from the manufacturer. Software and user specialists will also prepare and use their own. This should throw the whole field wide open.

To better understand the nature of microprogramming a no-order-set/no-data-structure computer, I believe it worthwhile to introduce a new word into our vocabulary: *firmware*. I use this term to designate microprograms resident in the computer's control memory, which specializes the logical design for a special purpose, e.g., the emulation of another computer. I project a tremendous expansion of firmware—obviously at the expense of hardware but also at the expense of software.

Microprogramming has, of course, been known since the early days of computing. The principle is simple—build a small micro-processor and drive it with a stored microprogram.

Take floating-point addition as an example. In the first generation, it was performed with a normally-programmed subroutine. In the second generation, it was implemented in circuitry. In third-generation microprogrammed computers, the floating-point addition is performed using a microprogram stored in read-only memory.

A noted software specialist, Mr. Opler is executive director of Computer Usage Education, Inc., in New York City. He was formerly director of programming systems for the parent company, Computer Usage Co.

One major factor making this possible is the ratio between main memory access time and circuitry speed. With integrated circuits (IC) and now LSI, times have plunged to 1-5 nanoseconds while the access time of main memory has slowly declined. This allows 30 to 70 micro-steps to be performed by circuitry between each main memory fetch or store. These 30 to 70 steps permit the implementation of third-generation order sets.

The memories are called read-only memories but, of course, the contents had to be written once in order to "load" the memory. Today this is done as part of the computer fabrication process. For the fourth generation, extension to slow-write fast-read (SW/FR) memory is anticipated.

The interface between "normal" programs and microprograms comes via the operation code. In effect, when each instruction is decoded, the operation code calls upon a specific microprogrammed "routine" stored in the micromemory.

To visualize the preparation of firmware, consider a special keypunch not available to software and user programmers. This punches 67-column cards (4 x 8 inches) with triangular holes (try that on your keypunch and card reader!). A special card reader loads decks of these 67-column cards into the fourth-generation SW/FR memory. This card reader probably will be locked so that only firmware specialists have access to it (we hope)!

what firmware can do now

Even in the third generation, firmware is expanding the capabilities of hardware and software. Its effect on reducing the cost of hardware is central to third-generation economics. Unprecedented production rates with reduced requirements for system checkout and delivery are due to both integrated circuitry and to microprogramming.

Specialized versions of standard product line computers have been prepared primarily by altering the microprogram in the read-only memory. The most spectacular success has been achieved with all-hardware compatibility via a special microprogram. The success of the IBM System/360 Model 30 in running IBM 1401 programs attests to this fact.

Emulation, a combination of software and hardware, operates by using two different microprogrammed features. Basically, an emulator is an interpretive simulator made much more efficient by using a microprogram to perform the basic interpreting loop—fetch an instruction, decode the address, access the contents of the address and perform the indicated operation. In addition, the op-codes that are most time- (or space-) consuming to simulate are directly executed by microprogramming those orders into the micromemory. Emulation is, of course, required when the micromemory is too small to contain the entire order set of the second-generation computer.

An interesting example of the power of extending existing capability by microprogramming is the evaluate (EVAL) instruction added to the IBM System/360 Model 50 delivered to Allen-Babcock Computing Inc. This command is reported to evaluate a PL/1 expression in standard form directly in hardware. Other added facilities include *floating decimal* instructions.

firmware in the fourth generation

Assuming the availability and accessibility of adequate SW/FR memories in the next generation, the entire *hardware/software* interface problems disappear only to be replaced by the more complex *hardware/firmware/software* interfaces. First, there is the hierarchy of components and corresponding responsibility.

Component	Responsibility	Organization
1. Circuitry	Hardware Designer	Manufacturer
2. Micromemory	Firmware Specialist	Manufacturer*
3. Control Programs	Software Specialist	Manufacturer*
4. Processors, Utilities, etc.	Software Specialist	Manufacturer*
5. Application programs	Application Specialist	User

*and/or software (firmware) producing organizations.

Secondly, firmware will assume a dominant role in structuring the computer. Manufacturers could (potentially) supply firmware decks allowing their fourth-generation computer to execute the order set of most popular third-generation computers–their own and competitors'. There could be a standard fourth-generation industry-wide order set and data structure.

But an even more significant impact would be felt by software. There is a general consensus that present software is too large, too complex and too slow. It is in the tradeoffs between software and firmware that most price/performance improvement should be obtained.

Third-generation computers require control programs to resolve and handle interrupts, to control multiprogramming and input/output dispatching, and to provide useful services such as resource allocation and protection, etc. The demands placed on control programs by the hardware

DATAMATION

Datamation 1955-1998

Datamation featured articles covering analysis, evaluation, implementation, and selection of technologies needed for companies to keep a competitive advantage.

https://archive.org/details/sim_datamation_1967-01_13_1/page/22/mode/2up

GitHub

- No debe confundirse con Git o GitLab.
  GitHub es una plataforma de desarrollo, de colaboración general, que se usa para alojar proyectos. Se utiliza principalmente para la creación de código fuente de programas de ordenador. Antes de enero de 2010, GitHub se llamaba Logical Awesome LLC. Los códigos que se almacenan en GitHub son de código abierto.

https://es.wikipedia.org/wiki/GitHub

Lua

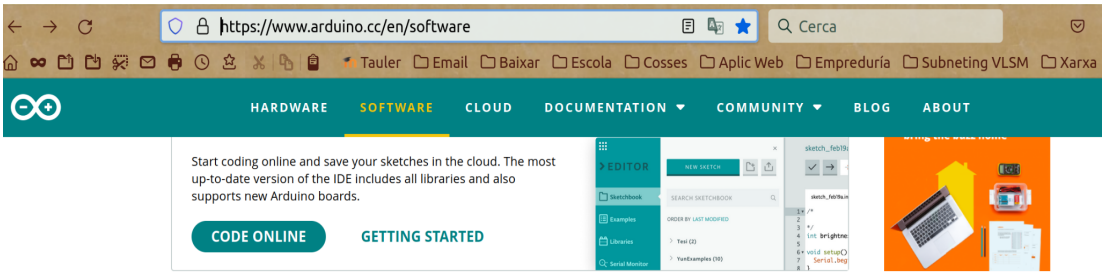
- Lua es un lenguaje de programación multiparadigma (orientado a objetos), imperativo, estructurado y bastante ligero, que fue diseñado como un lenguaje interpretado con una semántica extendible. Está diseñado principalmente para ser utilizado de manera incorporada en aplicaciones. Lua es un lenguaje multiplataforma y su intérprete está escrito en ANSI C. El nombre significa «luna» en portugués.

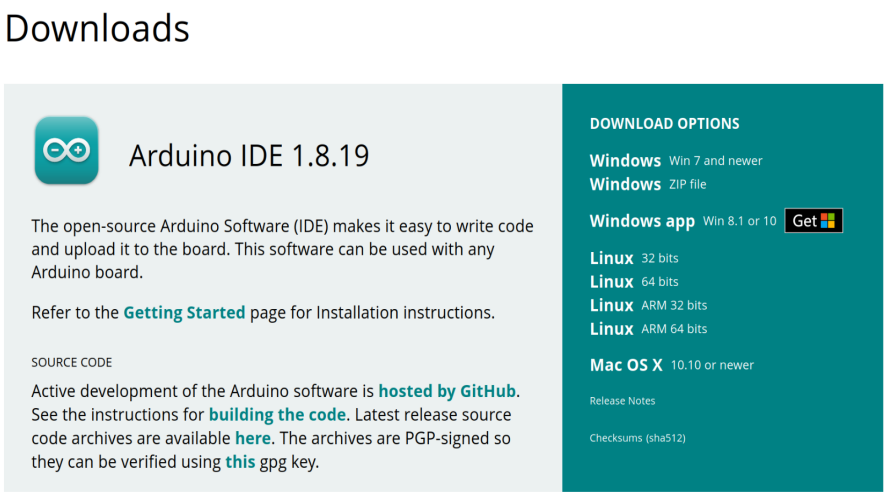
https://es.wikipedia.org/wiki/Lua

Instalación Arduino.

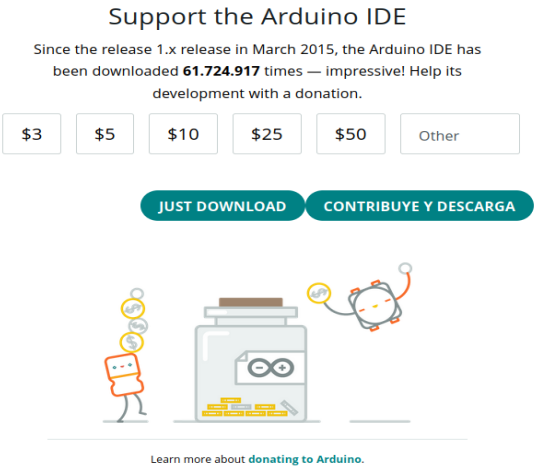
Para que nuestro pequeño proyecto funcione, hay que instalar unos programas, que intentaré explicar los pasos de una manera clara.

Nos dirigimos, via url, a https://www.arduino.cc/en/software, desde aquí seleccionamos la opción que nos interese más, ya sea por sistema operativo como por la arquitectura de nuestro ordenador (32 ó 64 bits). Yo elegí Linux de 64 bits.



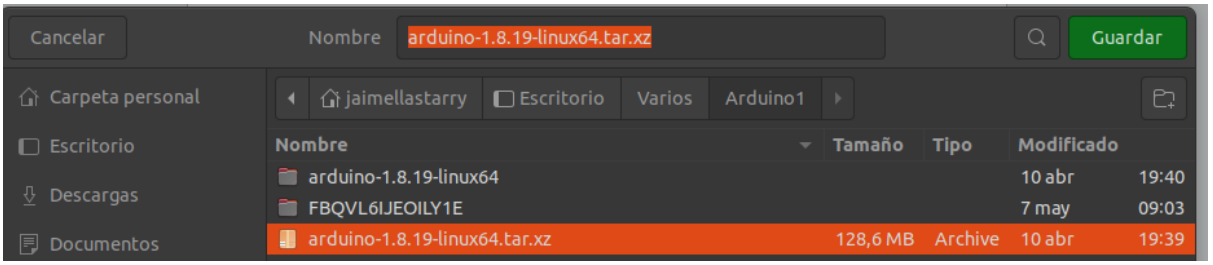


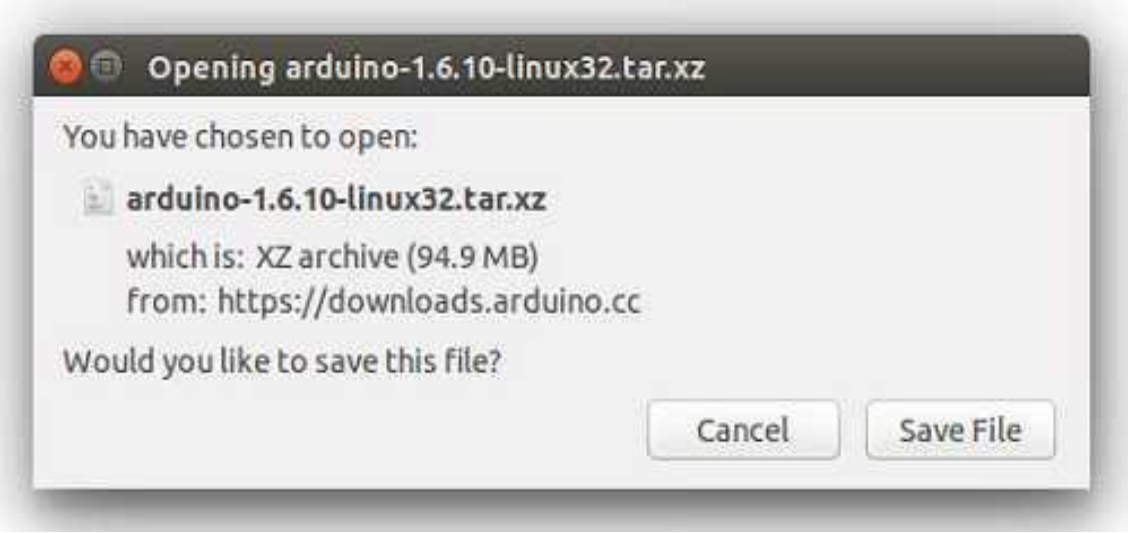
Acto seguido, al hacer clic en Linux de 64 bits aparece:

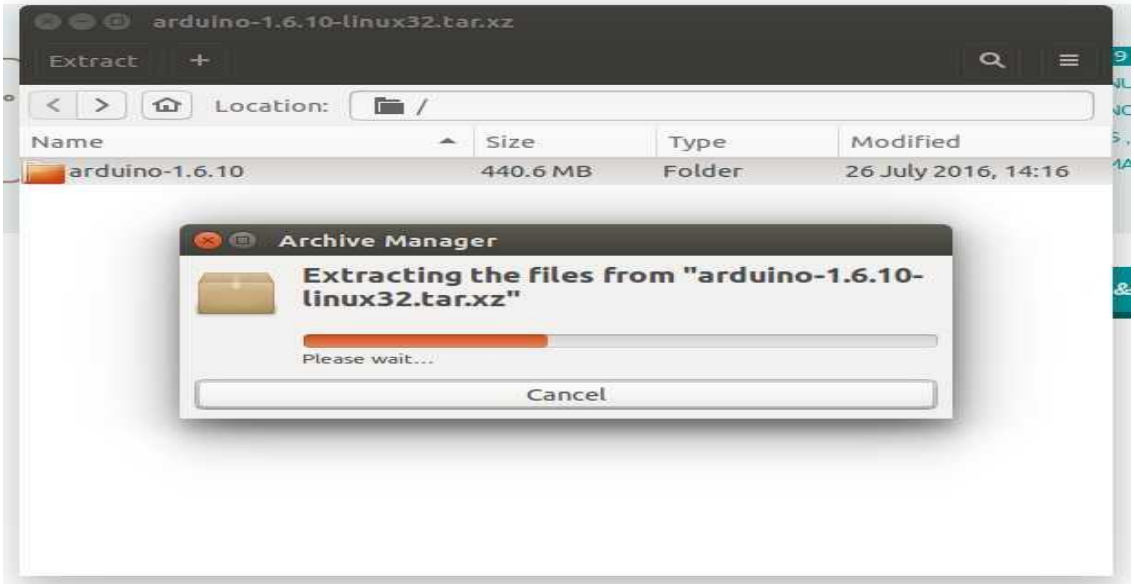


Al ser gratis, …¡Dona, que no cuesta nada! o Descarga.

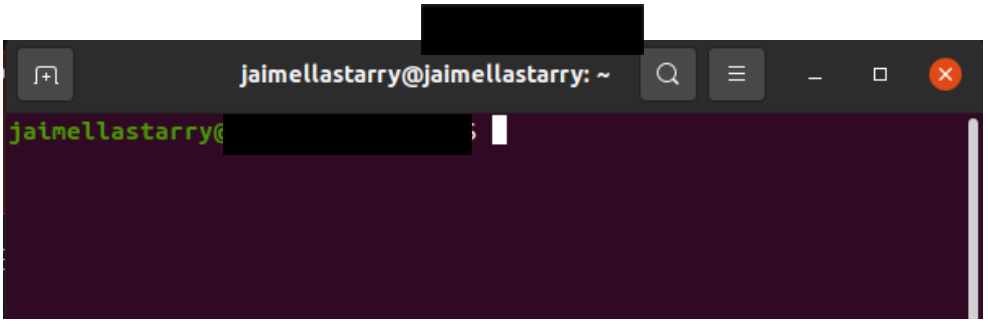
Al pulsar en Descargar, se nos abre una ventana de nuestro ordenador que nos indica dónde queremos descargar el archivo “arduino-1.8.19-linux64.tar.xz”. Clic en Guardar y descomprimimos el archivo acabado de descargar.



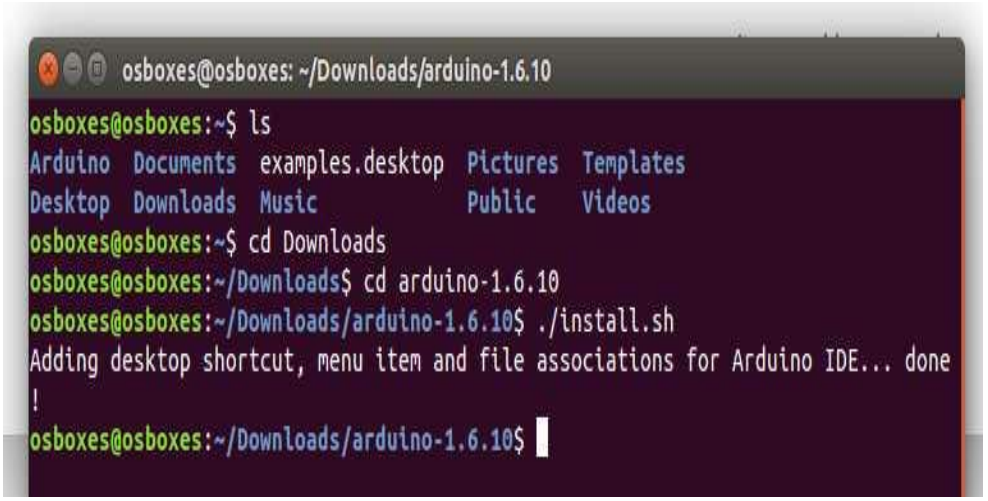




1. Cuando acabe de descomprimir, pasamos a ejecutar el programa. Se abre la carpeta dónde hemos descargado el programa, buscamos el archivo “**install.sh**”, hacemos clic derecho del ratón y pulsamos en la opción Ejecutar. Automáticamente se descarga el icono en el Escritorio y termina la instalación.
2. Si no entendemos la forma anterior de descargar, otra opción es:
    a. Abrir el Terminal.



    b. Listamos con “ls” y vemos las carpeta.



    c. Otro modo.

d. Ahora:

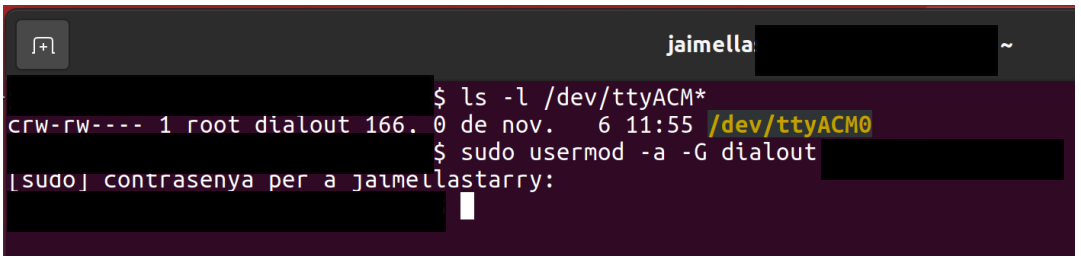
i. **ls -l /dev/ttyACM*** y nos devuelve algo así:

ii. **crw-rw---- 1 root dialout 188, 0 5 apr 23.01 ttyACM0** (El “0” al final de …**ACM** puede ser un número diferente. Lo más importante es el “**dialout**” que representa al grupo propietario.

iii. Ahora solo queda añadir nuestro usuario al grupo. Para ello, ponemos:

sudo usermod -a -G dialout <username>

Dónde sustituimos **username** por nuestro usuario.

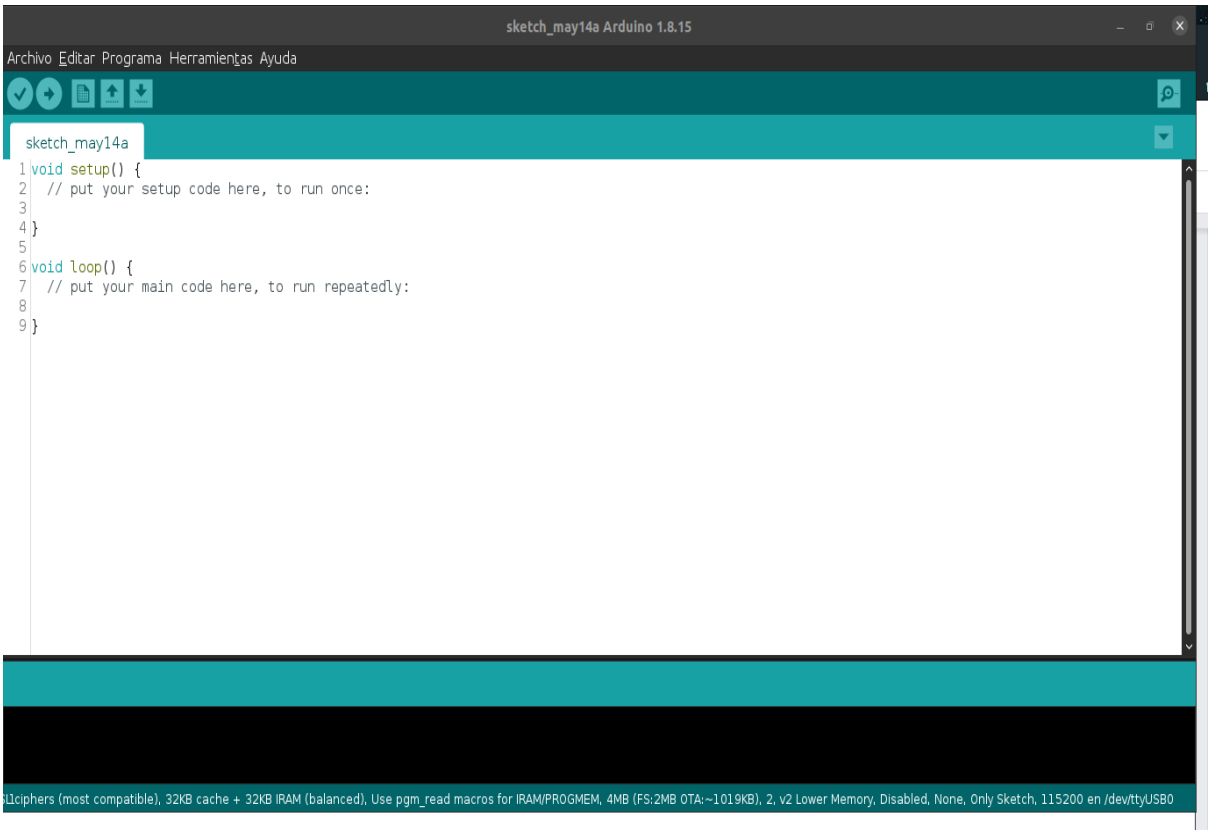


e. Ahora, reiniciamos el equipo y ya podremos acceder al puerto série del programa Arduino (IDE) sin errores.

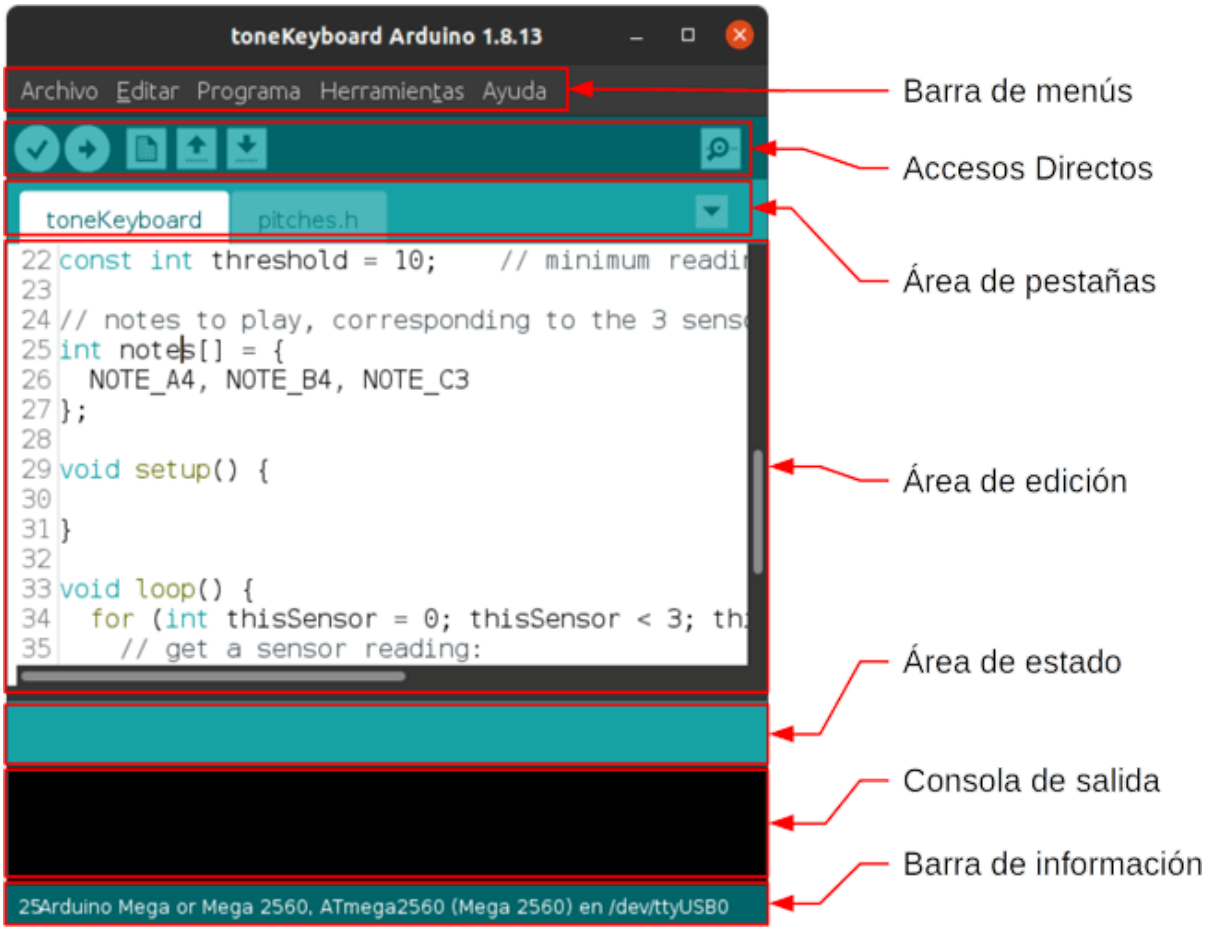
f. Ya tenemos instalado Arduino.

Configurar el IDE Arduino.

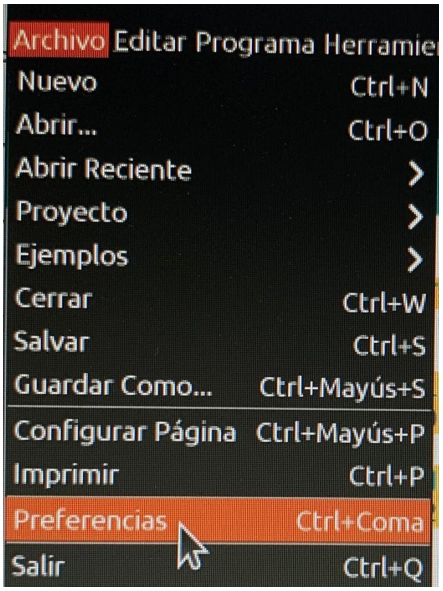
- Ventana de inicio IDE.
    - Iniciamos Arduino haciendo clic en el icono. Se abre el programa:



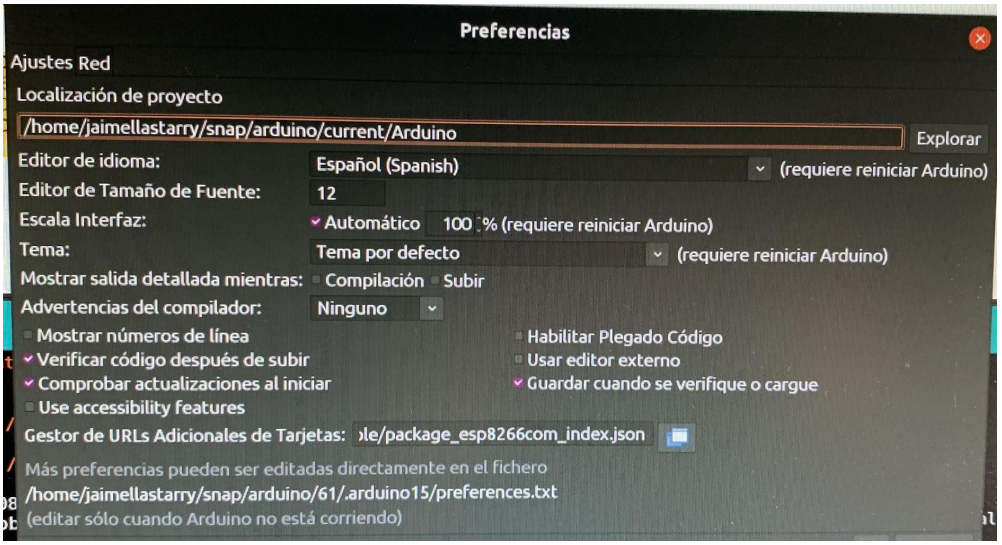
- Descripción de las partes de la ventana.



- Necesitamos instalar el complemento ESP8266. Para ello hacemos clic en Archivo > Preferencias.



- Aparece otra ventana.



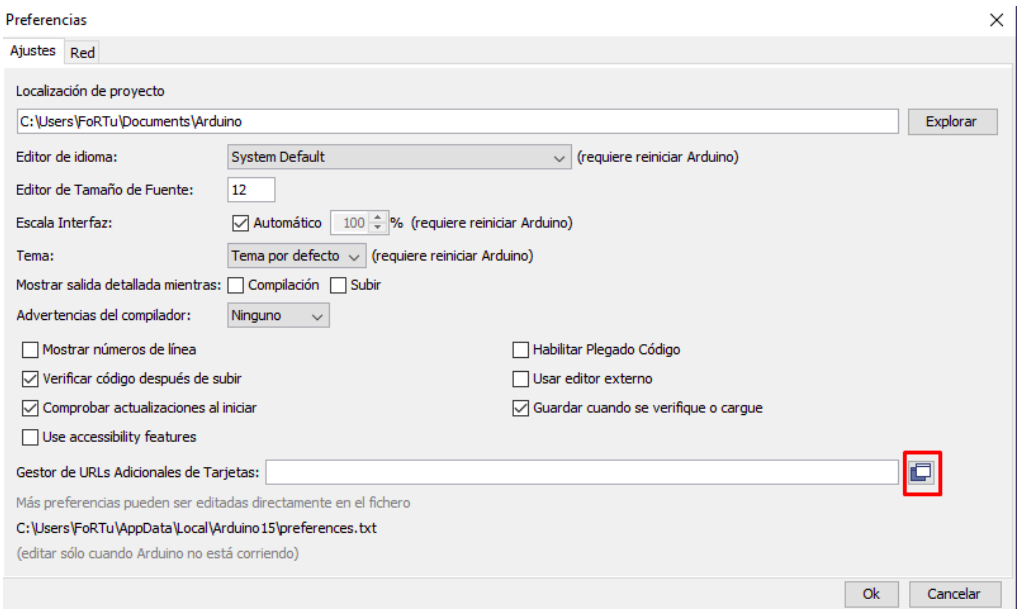
  - Aquí se ajustan los datos necesarios para configurar el arranque del IDE Arduino en cada inicio:

    - La carpeta dónde se ubica el proyecto.

    - Elegir el idioma.

    - Advertencias del compilador; esto quiere decir que, antes de ejecutar cualquier programa, realiza una verificación de la escritura y formato del programa. Recomiendo su activación.

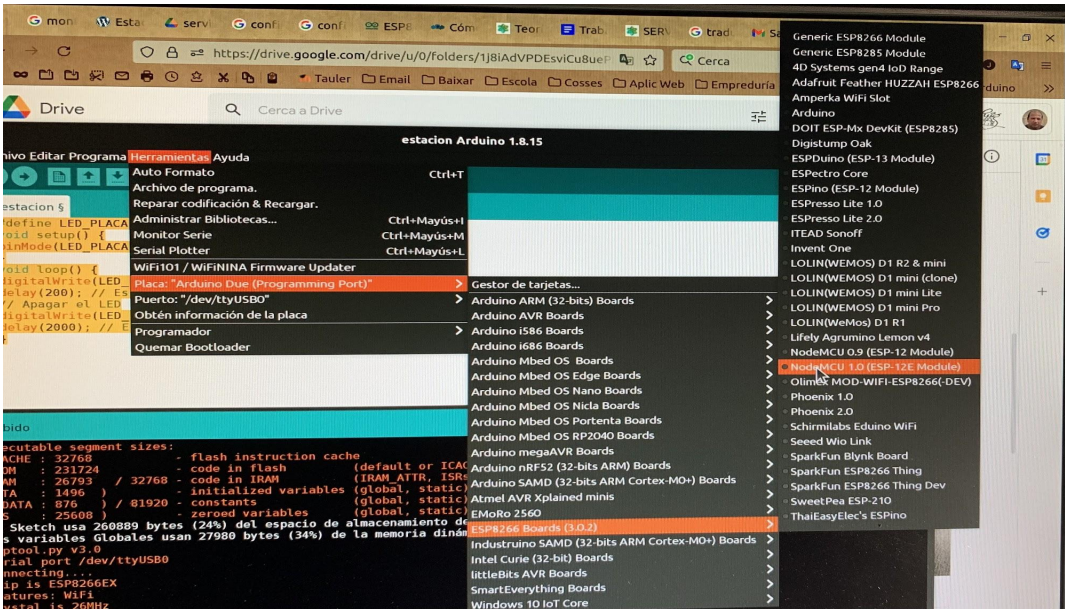
    - Gestor de URLs ….; Si se necesita una biblioteca de cualquier accesorio para tu proyecto, pones en tu buscador favorito, el nombre de lo que estás buscando (biblioteca esp8266) y aparecen muchas opciones de descarga, pero están mejor las de GITHUB (que es un portal creado para almacenar cualquier tipo de documento, código, etc. Pertenece a Microsoft). Para aplicar el código que necesitamos, clicamos en el cuadradito que hay a la derecha de la ventanita.



- Ahora, por ejemplo, añadimos esta línea, que es la biblioteca del ESP8266. http://arduino.esp8266.com/stable/package_esp8266com_index.json

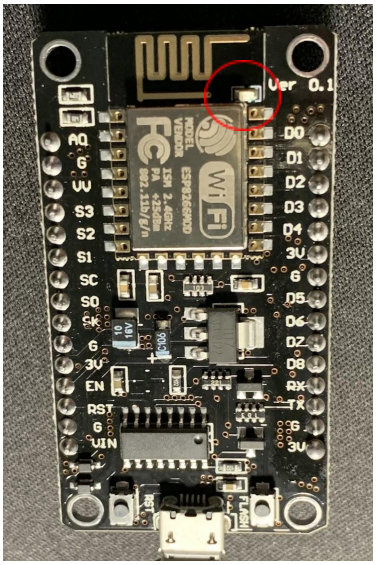
- Y pulsamos “Ok”, y nuevamente “Ok”.

- Ahora vamos a la pestaña Herramientas > Placas > Gestor de tarjetas. Este procedimiento puede variar dependiendo de la placa que utilicemos. Nosotros utilizamos la “NodeMCU 1.0 (ESP-12E Module)”, para seleccionarla, pulsamos en Placa > y se abrirá un desplegable con varias opciones. Buscamos la nuestra. Al verla, hacemos clic en ella.

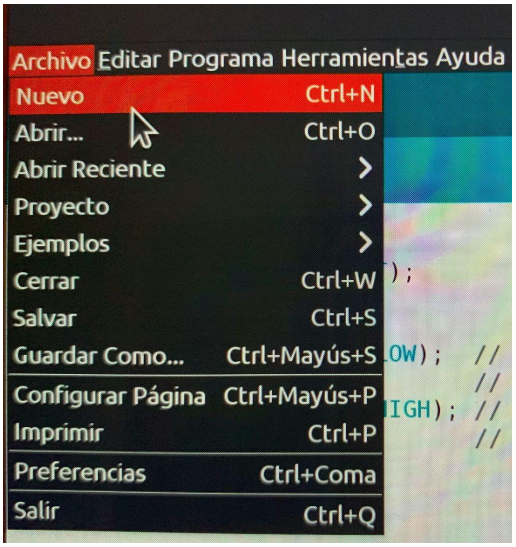


Prueba funcionamiento con placa ESP8266 V3 con IDE Arduino.

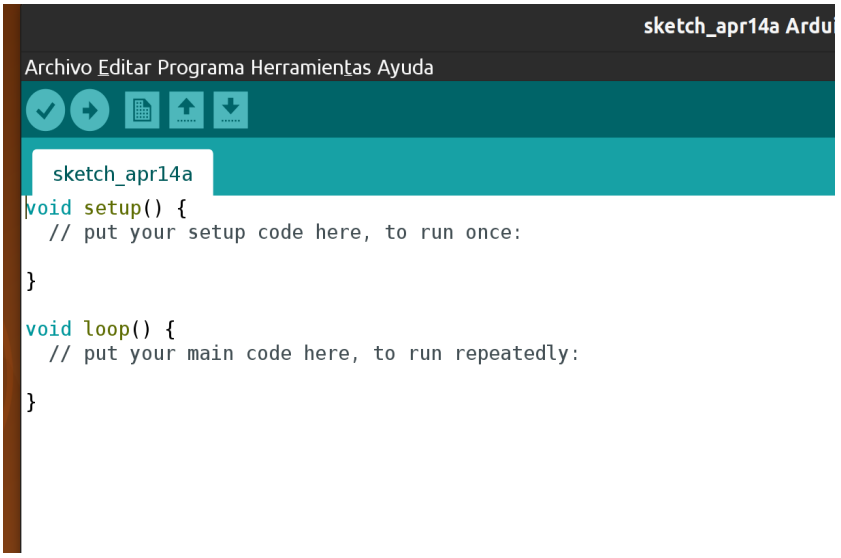
En este momento, se debe comprobar el funcionamiento de la placa Nodemcu ESP8266 V3. Para ello utilizaremos un programa, que se recomienda su uso antes de nada, para asegurarse del buen funcionamiento de la placa, mediante el Led que viene incorporado en ella.



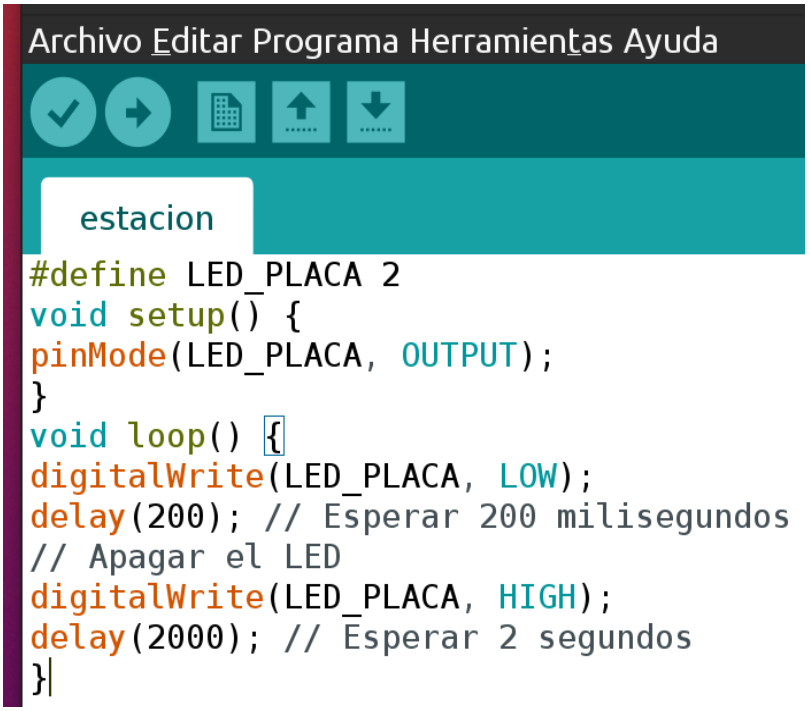
1. Abrimos IDE Arduino.
2. CLic en Archivo > Nuevo



3. Se abre un programa limpio.



4. Ahora empieza lo bueno: La Programación.

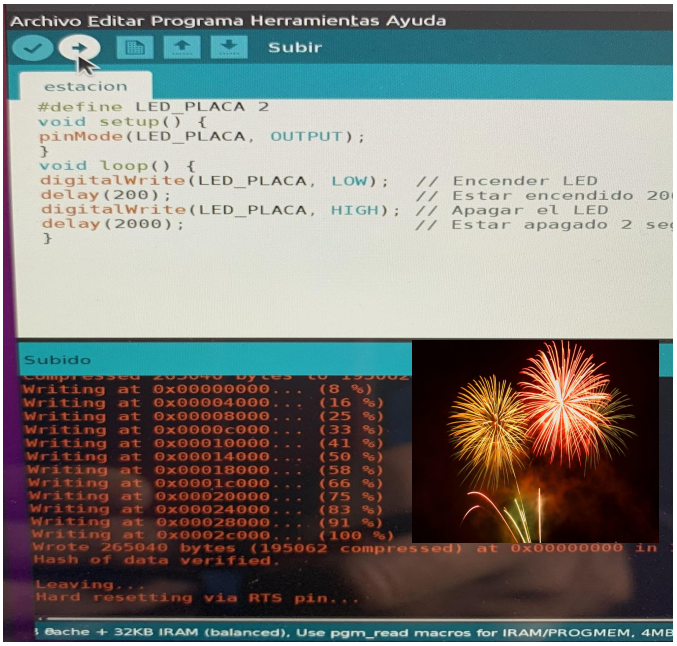


Quisiera comentar unas cosas:

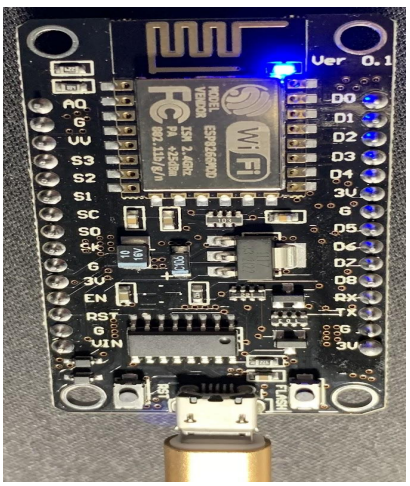
- ☐ Respecto al nombre del programa, éste se puede cambiar cuando se compila (círculo con flecha en la parte de accesos directos), o cuándo se sube (círculo con un Tick) y las partes del programa se explican en el punto siguiente del proyecto (programas).

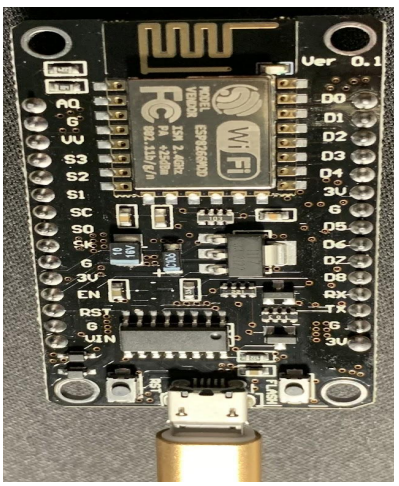
5. Acto seguido pueden suceder dos cosas:
   a. Que no esté bien el programa, con lo que nos saldrá un mensaje de error, parecido a este:

b. O por el contrario, que esté bién:



6. Dando como resultado, un encendido y apagado del Led integrado en la placa, el cual mediante unos pequeños retoques en el programa, podemos hacer que se enciendan más tiempo o, por el contrario, que esté apagado más tiempo, depende de lo que queramos hacer.





8 Anexo 2: Programas.

Robot.

Detector Paredes :

```
#include <AFMotor.h>
//Declaramos las variables de los pines
int Trig=12;
int Echo=13;
int ENA=5;
int ENB=6;
int IN1=3;
int IN2=4;
int IN3=2;
int IN4=7;
float cm;
float temp;

void setup() {
  Serial.begin(9600);
  pinMode(Trig, OUTPUT);
  pinMode(Echo, INPUT);

  pinMode(IN1,OUTPUT);
  pinMode(IN2,OUTPUT);
  pinMode(IN3,OUTPUT);
  pinMode(IN4,OUTPUT);

  pinMode(ENA,OUTPUT);
  pinMode(ENB,OUTPUT);


}

//Creamos un bucle que haga que cuando detecte algo a menos de 30cm haga las
//funciones back y left, tambien se puede back y right
void loop() {

  digitalWrite(Trig, LOW);
  delayMicroseconds(2);
  digitalWrite(Trig,HIGH);
  delayMicroseconds(10);
  digitalWrite(Trig, LOW);
```

```
//Decimos, si la distancia es menos que 10 echa para atrás hasta que hayan 30cm y
//gira a la izquierda y cuando haya más de 40cm vez hacia delante.
  temp = float(pulseIn(Echo, HIGH));
  cm = (temp * 17 )/1000;
   if (cm < 30 && cm > 10)
   {
          back();
          delay(1000);
          Left();
          delay(500);

          }
   if (cm >=40)
   {
          forward();
          delay(100);

          }


          if (cm < 10)
   {
          STOP();
    // delay(100);

          }
  Serial.print("Echo =");
  Serial.print(temp);
  Serial.print(" | | Distance = ");
  Serial.print(cm);
  Serial.println("cm");
  delay(100);

}

//Declaramos la función back y le decimos donde queremos corriente en los pines
void back(){

  analogWrite(ENA,170);
  analogWrite(ENB,170);

  digitalWrite(IN1,LOW);
```

```
  digitalWrite(IN2,HIGH);
  digitalWrite(IN3,HIGH);
  digitalWrite(IN4,LOW);
  Serial.println("Forward");
}

//Declaramos la función forward y le decimos donde queremos corriente en los pines
void forward(){
analogWrite(ENA,170);
  analogWrite(ENB,170);
  digitalWrite(IN1,HIGH);
  digitalWrite(IN2,LOW);
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,HIGH);

  Serial.println("Back");
}


void Left(){
  analogWrite(ENA,200);
  analogWrite(ENB,200);
  digitalWrite(IN1,HIGH);
  digitalWrite(IN2,LOW);
  digitalWrite(IN3,HIGH);
  digitalWrite(IN4,LOW);

  Serial.println("Left");
}

//Declaramos la función right y le decimos donde queremos corriente en los pines
void Right(){
  analogWrite(ENA,170);
  analogWrite(ENB,170);
  digitalWrite(IN1,LOW);
  digitalWrite(IN2,HIGH);
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,HIGH);

  Serial.println("Right");
}
//Declaramos la función stop y le decimos donde queremos corriente en los pines
void STOP(){
```

```
  digitalWrite(ENA,LOW);
  digitalWrite(ENB,LOW);
  digitalWrite(IN1,LOW);
  digitalWrite(IN2,LOW);
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,LOW);
  Serial.println("STOP");
}
```

Medidor:

```
//----------------Bibliotecas-------------------
#include <ESP8266WiFi.h>          // Conectarse vía wifi
#include <ESP8266WebServer.h> // Configurar servidor y solicitudes HTTP
#include <DHT.h>              // DH11
#include <BH1750.h>           //GY-30
#include <Wire.h>             //GY-30

//----------------- Tipo de sensor-----------

//------SENSOR DHT11--------------------------
//Pins DHT11
//VCC = 3v
//GND = G
//DATA = PIN D4
#define DHTTYPE DHT11   // DHT 11 - #define es un valor constante

//------SENSOR GY-30--------------------------
//Pins GY-30
//VCC = 3V
//GND = G
//SCL = PIN D2
//SDA = PIN D3

//-----------------------------------------------

/*Nombre wifi y contraseña*/
// Añadir los datos de vuestra red
//const char* ssid = "----------";
// const es como una variable pero no se debe cambiar su valor
//const char* password = "---------";

//Iphone Jaime
const char* ssid = "iPhone de Jaume";
const char* password = "t8aa4jsory61z";

//wifi instituto
//const char* ssid = "SuperXEILL";
//const char* password = "";
```

```
//Servidor----------

ESP8266WebServer server(80);//Declaramos webserver, y abrimos el puerto
//80, predeterminado para HTTP

// DHT Sensor
#define DHTPin       2 //Definimos el pin conectado al sensor
//uint8_t DHTPin = D2;

//----------Inicio de los sensores
//-----DHT11------
DHT dht(DHTPin, DHTTYPE);
//-----GY-30------
BH1750 lightMeter;

//Variables-------
float Temperature;//float son variables
float Humidity;
float Heatindex;
float LightMeter;

// aquí ponemos los comandos de la subrutina
void setup() { // Configurar servidor HTTP antes de ejecutarlo

Serial.begin(9600); // Velocidad de Baudio, entrada del IDE Monitor Serie
delay(100);             // Tiempo de lectura del Baudio

//----DHT11-------
pinMode(DHTPin, INPUT); //Usaremos el pin 2 como entrada.
dht.begin();            //begin=Inicializamos el objeto

//----GY-30------
Wire.begin(D2,D3);//begin=Inicializamos el objeto y definimos los pines
lightMeter.begin();//begin=Inicializamos el objeto

// Conexión a Wifi----------
Serial.println("Conectando a  ");
Serial.println(ssid);

WiFi.begin(ssid, password);//Aqui carga los datos ssid y password

while (WiFi.status() != WL_CONNECTED) {// Wifi.status=Mientras nos
//conectamos, verificamos el estado de conectividad
```

```
delay(1000);
Serial.print(".");
}
Serial.println(""); //Una vez conectados imprime mediante wifi.localIP el valor
//de la IP
Serial.println("WiFi conectada..!");
Serial.print("Copiar IP en el URL: ");  Serial.println(WiFi.localIP());

server.on("/", handle_OnConnect);// Llamar a la funcion handle_OnConnect
//cuando se solicite URL o /
server.onNotFound(handle_NotFound);//Llamar a la función handleNotFound
//cuando se solicite otra cosa que no sea / o URL

server.begin(); //Iniciar servidor
Serial.println("HTTP servidor conectado");

}
// Aquí se inicia el bucle loop
void loop() {

server.handleClient(); //Escuchar las solicitudes de los clientes

}

void handle_OnConnect() {

Temperature = dht.readTemperature(); // Obtener valores temperatura
Humidity = dht.readHumidity(); // Obtener valores humedad
Heatindex = dht.computeHeatIndex(Temperature, Humidity, false);// Obtener
//valores Indice calor
LightMeter = lightMeter.readLightLevel(); // Obtener valores lumens
server.send(200, "text/html",
SendHTML(Temperature,Humidity,Heatindex,LightMeter));/*Enviar estado
HTTP, 200=Bien, y mandar el
texto al cliente, en este caso, las lecturas de los sensores*/
delay(100);
}

void handle_NotFound(){
server.send(404, "text/plain", "Not found");/*Enviar 404:Not found cuando no
haya conexión con el servidor o controlador */
}
```

```
//Declarar String= Toma los valores para generar dinámicamente el contenido
//HTML
String SendHTML(float Temperaturestat,float Humiditystat,float Heatindexstat,
float LightMeterstat){
String ptr = "<!DOCTYPE html> <html>\n"; //Se está enviando código HTML
ptr +="<head><meta name=\"viewport\" content=\"width=device-width,
initial-scale=1.0, user-scalable=no\">\n";
/*La línea superior hace que la página web se vea en cualquier navegador*/
ptr +="<title>Mediciones vía ESP8266</title>\n";//Título pestaña página web
ptr +="<meta charset=utf-8></head>";//Tipo de codificación reconocida por
//Unicode
ptr +="<meta http-equiv='refresh' content='10'/>";//Refresca la ventana cada
//10 segundos
ptr +="<div id=\"webpage\">\n";//Establece el encabezado
ptr +="<h1>VEHÍCULO CON MEDICIÓN AMBIENTAL MÓVIL</h1>\n";//Título
//Principal
ptr +="<h2><font color='#009900'>Proyecto de Jaime Llastarry y Marko
//Pareja</font></h2>";//Subtítulo
ptr +="<h3>SMIX2B 2021-2022</h3>\n";
ptr +="<br>";//Espacio entre líneas
ptr +="<style>html { font-family: Helvetica; display: inline-block; margin: 0px
auto; text-align: center;}\n";
ptr +="body{background-color: #0aeaed;margin-top: 50px;} h1 {color:
#047ac7;margin: 50px auto 30px;}\n";
ptr +="p {font-size: 24px;color: #820303;margin-bottom: 10px;}\n";
ptr +="</style>\n";
ptr +="</head>\n";
ptr +="<body>\n";
/*Con Style usamos algo de CSS para diseñar la apariencia. Elegimos la
fuente Helvetica, definimos el contenido que se mostrará como un bloque en
línea y alineado en el centro.
Desde body hasta /style, establece el color de la fuente, el margen alrededor
del cuerpo y las etiquetas h1 y p*/

ptr +="<p>Temperatura: ";//Nombre que aparece en la web
ptr +=(int)Temperaturestat -1;//Imprime la lectura, añado la diferencia
//respecto a otro medio de lectura
ptr +=" ºC</p>";//Nombre que aparece en la web
ptr +="<p>Humedad: ";
ptr +=(int)Humiditystat +8;/*Se llama a la variable, y nos devuelve los valores
enteros mediante la conversión de tipo llamada HTML &deg */
ptr +=" %</p>";
ptr +="<p>Índice de calor: ";
```

```
ptr +=(int)Heatindexstat ;
ptr +=" ºC</p>";
ptr +="<p>Luminosidad: ";
ptr +=(int)LightMeterstat +150;
ptr +=" Lumens</p>";
ptr +="</div>\n";//Fin etiqueta div
ptr +="</body>\n";//Fin cuerpo
ptr +="</html>\n";//Fin página web
return ptr; // iniciar  ptr
}// volver al principio, String SendHTML
```

9 Manual de usuario del Medidor Ambiental.

Medidor Ambiental

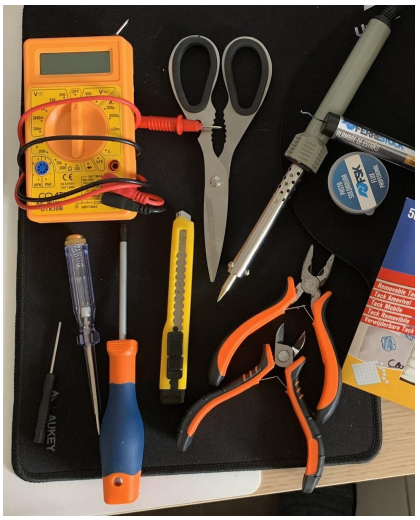
Jaime Llastarry Jansana
Tutores: Rocco Ballester y Jordi Farrero
CCFF SMIX2
Institut Puig Castellar, a 31 de mayo de 2022

Índice 2

Manual de usuario: Kit de Medición Ambiental.

Kit Medición.

A continuación, les dejamos una imagen con las herramientas que posiblemente se necesiten para montar este kit.



- Destornilladores: estrella y plano (pequeños).
- Alicates: planos y de corte.
- kit de soldadura (sin experiencia, mejor no utilizarlo).
- Tester (para fallos eléctricos).
- Tijeras.
- Cutter.

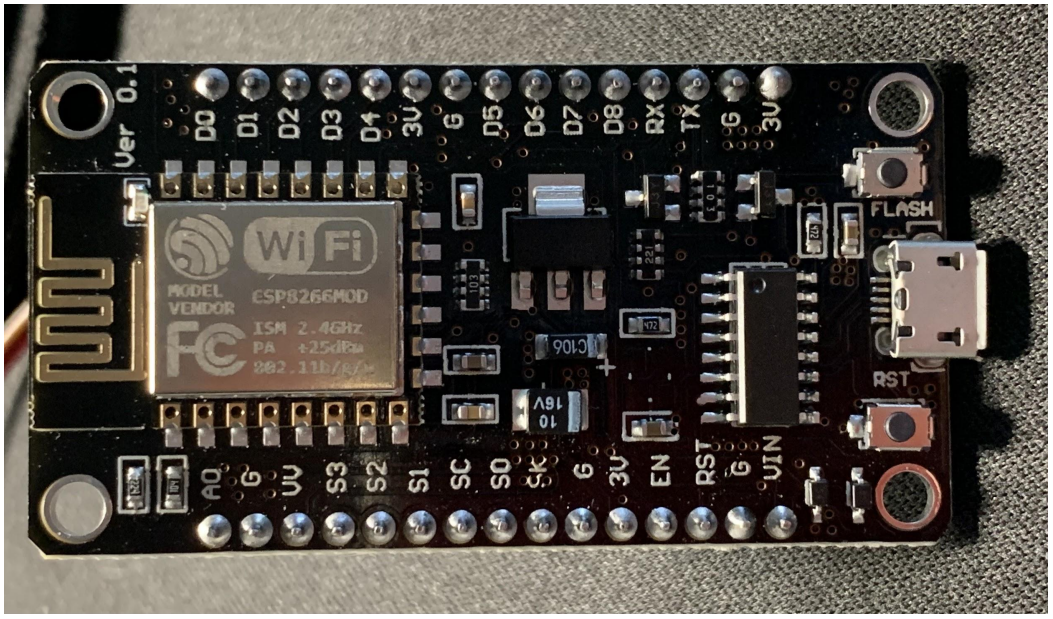
Partes.

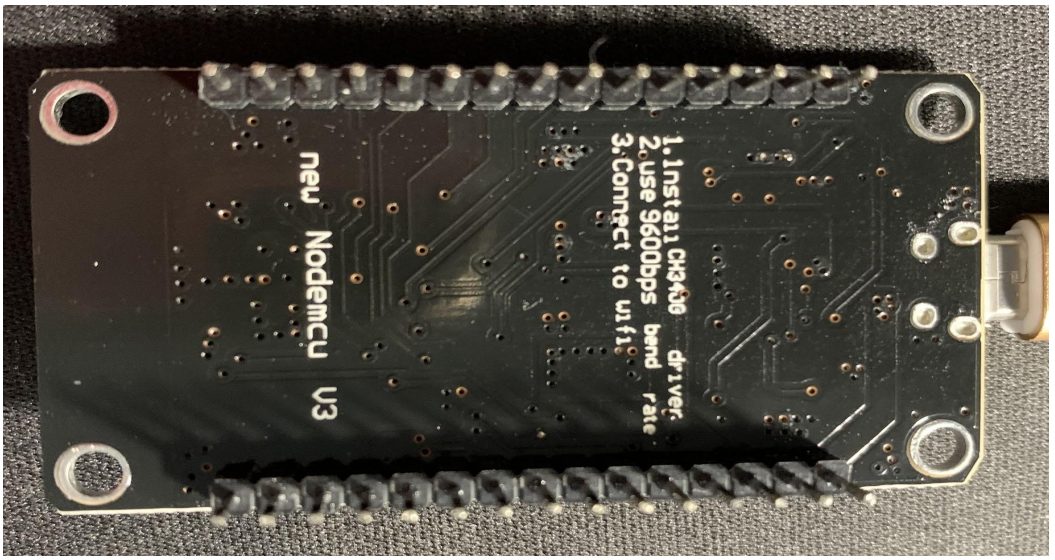
Nodemcu ESP8266.

Este tipo de placa incorpora componentes que nos ayudan a programar y conectar el módulo a nuestros circuitos.

NodeMCU y sus características principales:

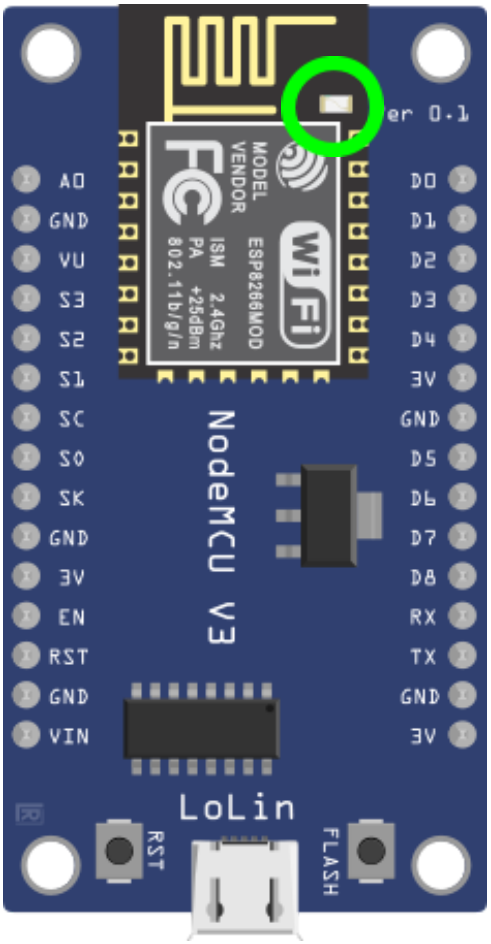
- Conversor Serie-USB para poder programar y alimentar a través del USB
- Fácil acceso a los pines
- Pines de alimentación para sensores y componentes
- LEDs para indicar estado
- Botón de reset





LED integrado dentro de la placa.

El NodeMCU V2 tiene un LED integrado. Está asociado al pin D0 (GPIO 16), y hará que, utilizando un programa, podamos encender y apagar el LED.

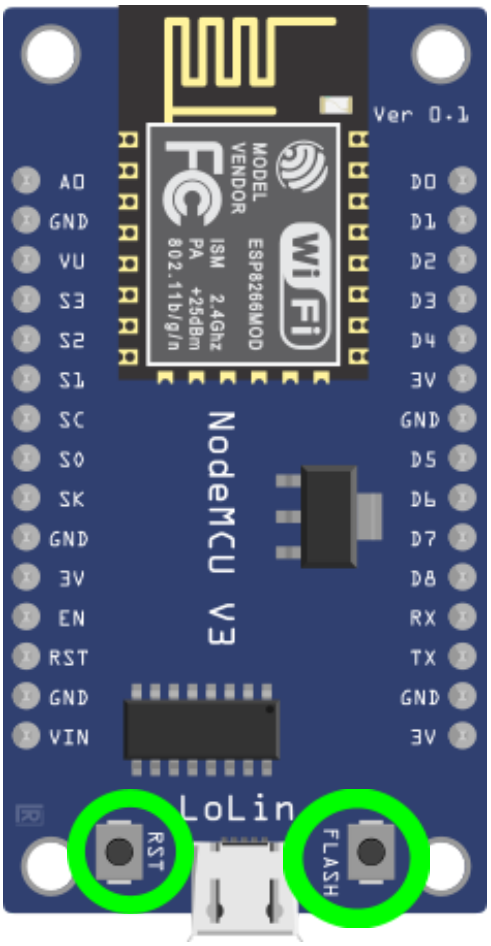


Éste LED también se enciende cada vez que la placa:

- Se conecta la alimentación.
- Envía o recibe datos.
- Si por programa, se le ordena.

Pulsadores integrados dentro de la placa.

La placa cuenta con dos pulsadores. El botón de RESET (RST) que si lo pulsamos se resetea la placa y comienza la ejecución de cero, y otro botón de FLASH, que ordena cargar el programa sin resetear.



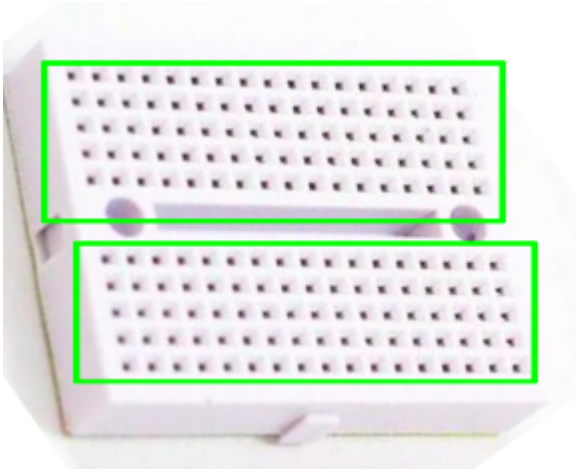
El botón RST (Reset) hace eso, resetear. Esto no quiere decir que elimine el código, lo único que hace es reiniciar la ejecución del programa, pasando por la función *setup()*.

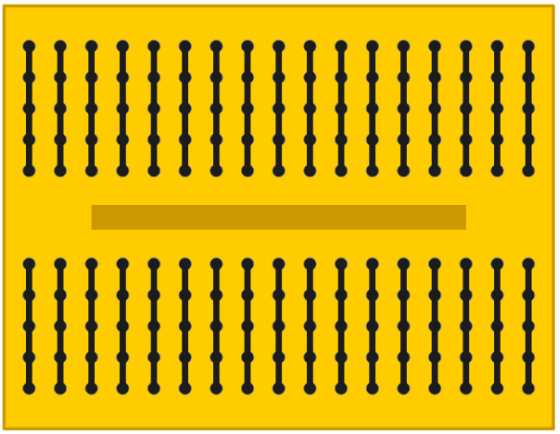
El botón de FLASH nos permite cargar un programa o firmware. Esto no es único de NodeMCU o del ESP8266, todos los microcontroladores tienen como mínimo dos estados.

- El estado de ejecución es cuando el microcontrolador ejecuta el programa que hemos cargado.
- El estado carga de programa o de firmware nos permite subir un programa al microcontrolador.

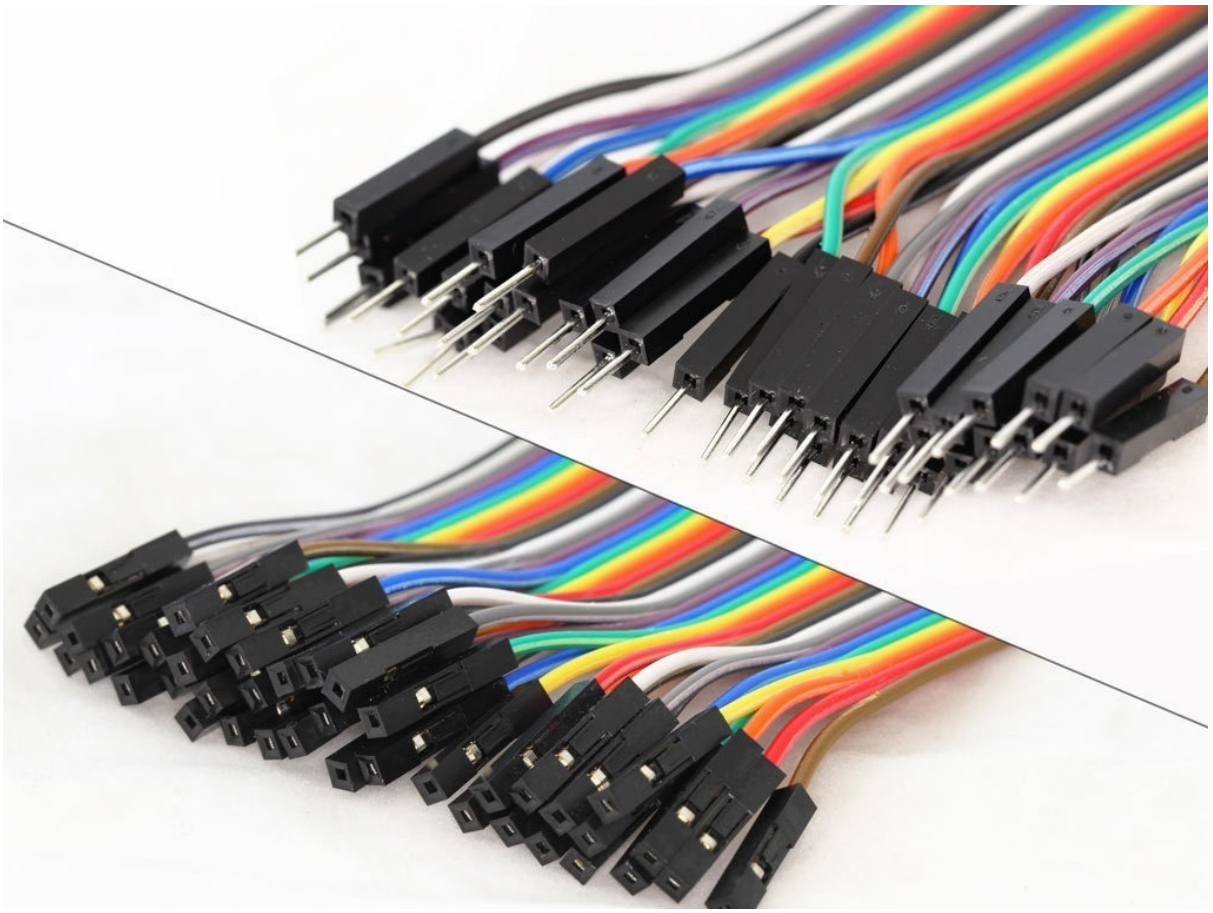
Protoboard.

Una Protoboard o breadboard es una tabla rectangular de plástico con muchos y pequeños agujeros en ella. Estos agujeros permiten insertar fácilmente componentes electrónicos para hacer un proyecto o un circuito electrónico, como por ejemplo, el que tenemos entre manos. Aquí es dónde los pines deben estar conectados firmemente y en la posición adecuada, ya que si no es así, no funciona nada del proyecto. La conexión se puede hacer en uno de los dos lados de 5 columnas y 17 filas. Las conexiones internas están en columnas, es decir, del 1º al 5º pin.





Cableado.



Este tipo de cableado se utiliza para realizar conexiones entre los componentes del kit. Son sencillos y limpios en las conexiones, ya que no hay que soldarlos, ni pegarlos con ningún tipo de producto.

Los cables pueden ser de varios tipos de conexión:

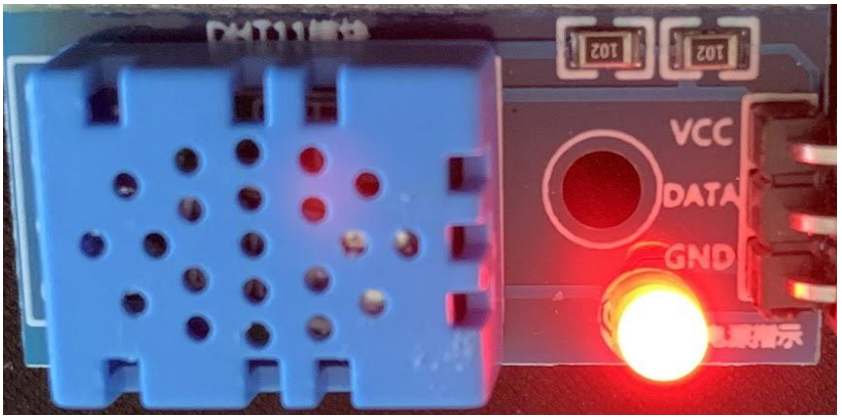
- Hembra-Macho (como en la imagen).
- Hembra-Hembra.
- Macho-Macho.

Más adelante veremos su utilización en varios componentes.

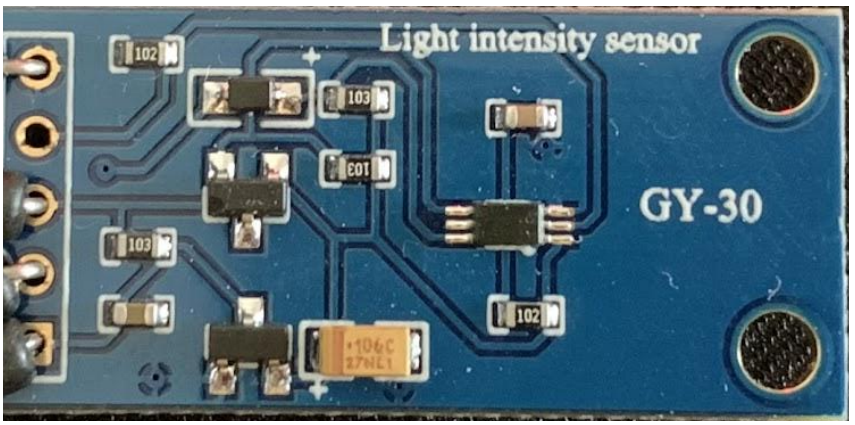
Sensores.

Un sensor es un dispositivo capaz de captar acciones, o estímulos externos, y responder a ellos. Los sensores miden las magnitudes físicas y las transforman en señales eléctricas capaces de ser entendidas por un microcontrolador. Los sensores son utilizados para el desarrollo de interfaces físicas, sistemas robóticos y otros proyectos. En este kit vienen incluidos:

Sensor DHT11 (mide la temperatura y la humedad)



Sensor GY-30 (mide la luminosidad del ambiente)



Son fáciles de reponer o sustituir por otros. No necesitan ningún tipo de mantenimiento y por su calidad-precio, son muy buena solución para este tipo de kits.

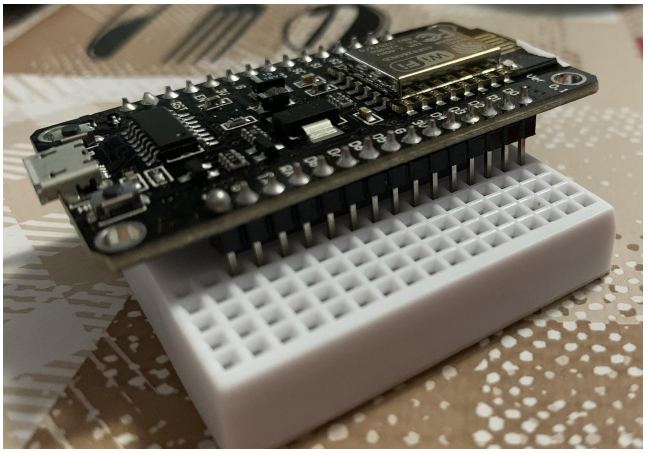
Proceso de Montaje.

Es muy recomendable seguir un orden para montar éste tipo de kits. Vamos, mediante este manual, a explicar los pasos a seguir, teniendo en cuenta que el tema de programación, para su correcto funcionamiento, está en el **Anexo 1: Información e instalación programas, página 58** de este proyecto.

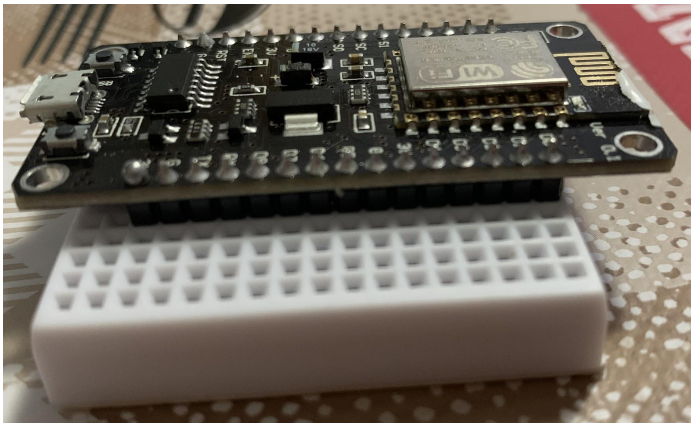
Paso 1. Comprobar que, las piezas internas y externas del envase, estén en perfectas condiciones: los sensores y placas vienen envasadas al vacío para proteger sus componentes electrónicos.

Paso 2. Trabajar en un espacio limpio de polvo y lejos de posibles zonas electroestáticas, para proteger el buen funcionamiento de dichas piezas.

Paso 3. Mediante imágenes, intentaremos ser lo más claros posibles en su montaje: Nodemcu ESP8266 a protoboard

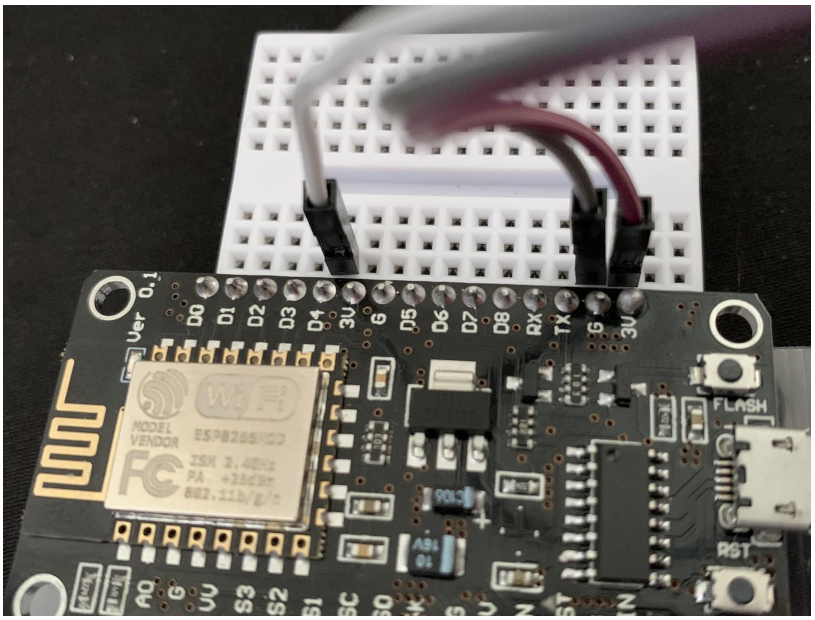


En esta imágen observamos, con atención, su posicionamiento, ya que es recomendable colocarlo de la misma manera para seguir el procedimiento correcto y el buen funcionamiento de los componentes de este kit.



Mediante una leve presión, acabamos de insertar los pines en la protoboard.

DHT11

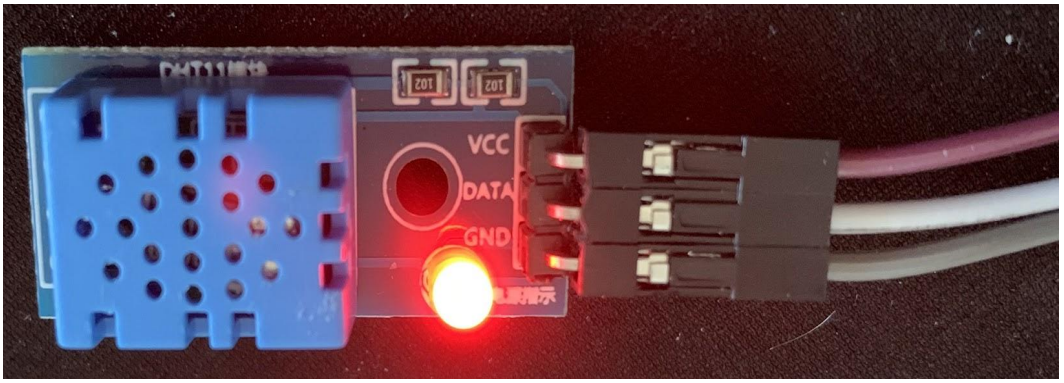


¡¡A TENER EN CUENTA!!

Los cables van conectados siempre en la misma línea.

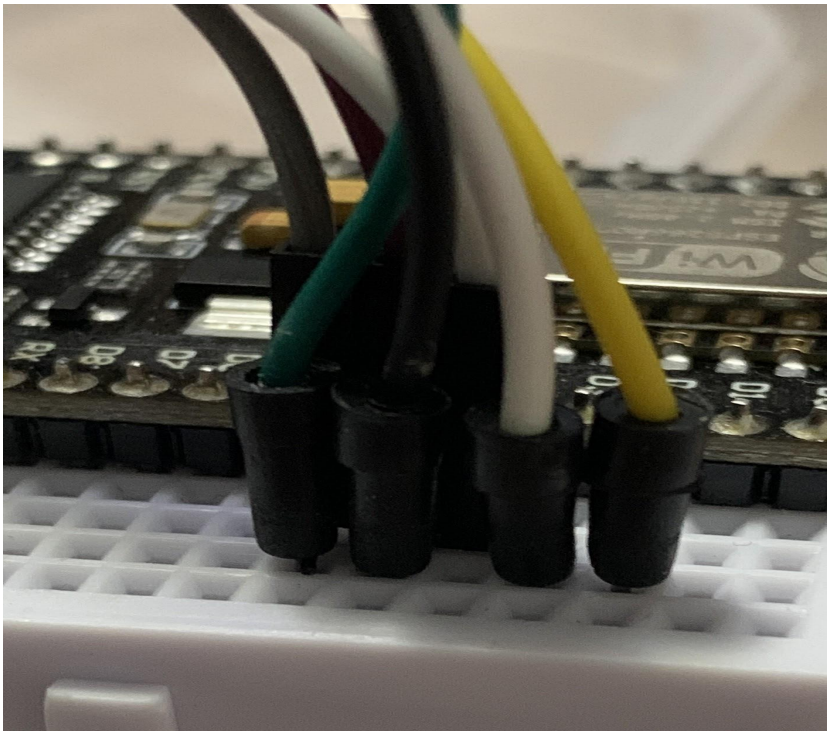
En la imagen se observa que los cables van en una posición concreta:

- ➔ VCC = 3v (color lila, tipo macho en la protoboard y hembra en el sensor). Puede ponerse en varias ubicaciones.
- ➔ GND = G (color gris, tipo macho en la protoboard y hembra en el sensor). Puede ponerse en varias ubicaciones.
- ➔ DATA = D4 (color blanco, tipo macho en la protoboard y hembra en el sensor). Esta posición puede variar, al igual que los colores de los cables, pero se ha de tener en cuenta que si se varía la posición, también se tienen que variar los datos en el programa que se va a utilizar, cosa no recomendable.



La luz se encenderá cuando se conecte todo al ordenador.

GY-30

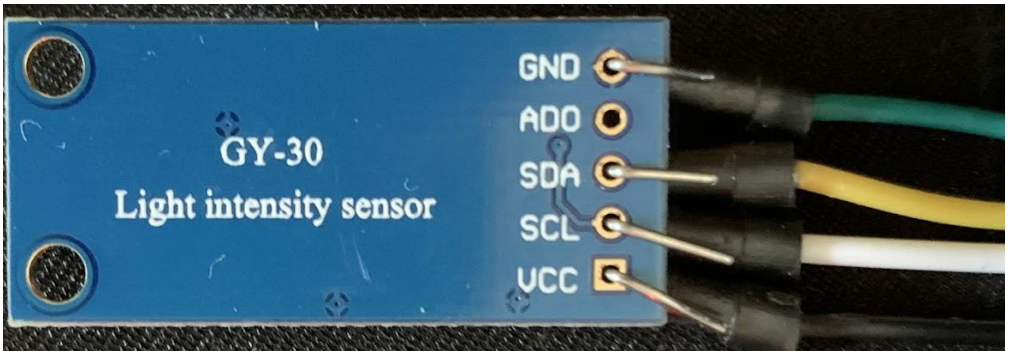


¡¡A TENER EN CUENTA!!

Los cables van conectados siempre en la misma línea.

En la imagen se observa que los cables van en una posición concreta:

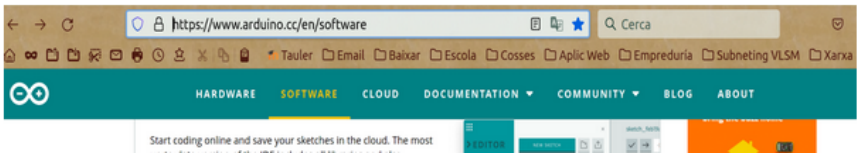
- ➔ VCC = 3v (Color negro, tipo macho en la protoboard y macho en el sensor). Puede ponerse en varias ubicaciones.
- ➔ GND = G (Color verde, tipo macho en la protoboard y macho en el sensor). Puede ponerse en varias ubicaciones.
- ➔ SDA = D2 (Color amarillo, tipo macho en la protoboard y macho en el sensor). Es la parte de datos.
- ➔ SCL = D3 (Color blanco, tipo macho en la protoboard y macho en el sensor). Es la parte del reloj interno.



Paso 4. Mediante el **Anexo 1: Información e instalación programas, página 58**, de este proyecto, seguir los pasos de instalación de los programas necesarios para el funcionamiento del kit.

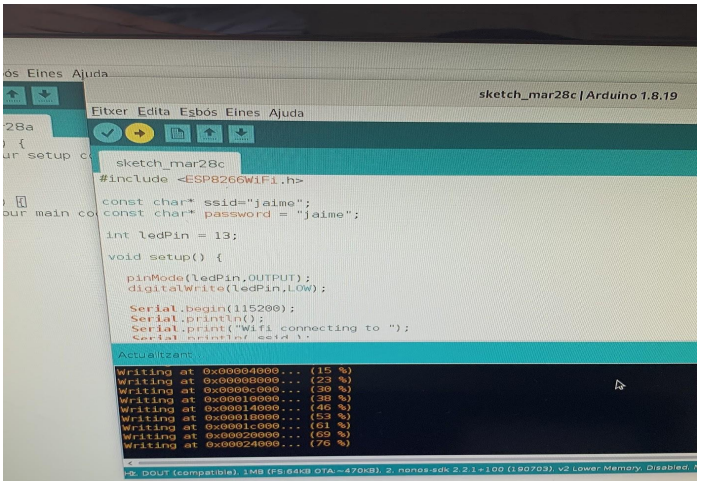
Instalación Arduino.

Nos dirigimos, via url, a https://www.arduino.cc/en/software, desde aquí seleccionamos la opción que nos interese más, ya sea por sistema operativo como por la arquitectura de nuestro ordenador (32 ó 64 bits). Yo elegí Linux de 64 bits.



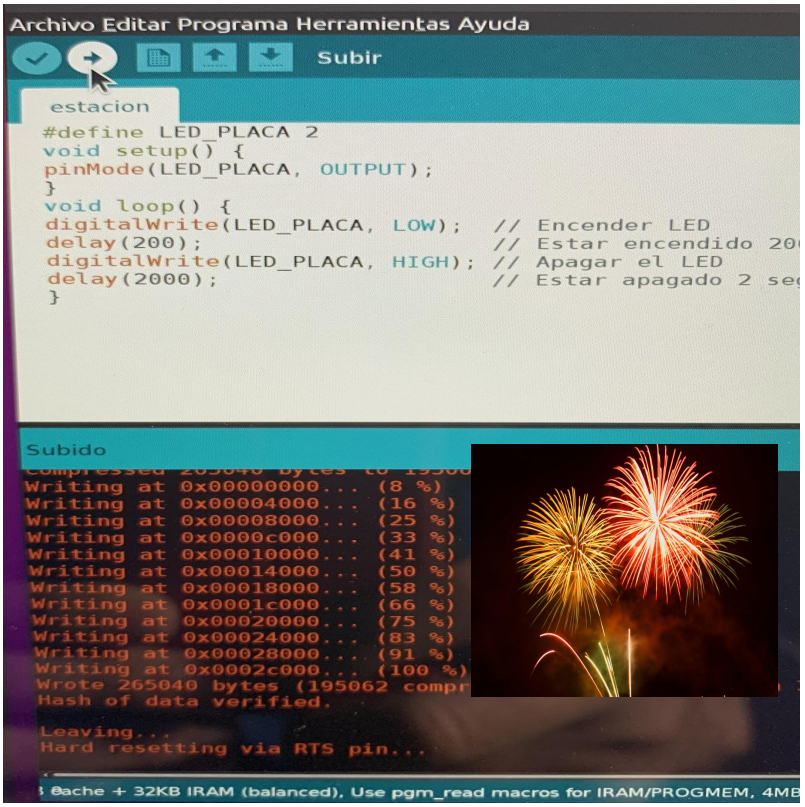
Paso 5. Llega el momento de la verdad: Probarlo todo junto.

1. Conectar el kit al ordenador, con el programa en pantalla.

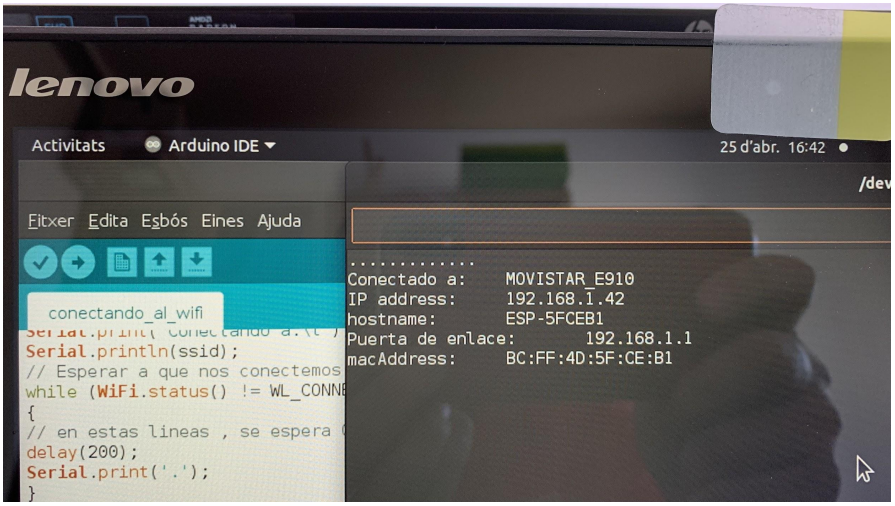


2. Hacer clic en el botón de subir el programa Arduino, en la imagen superior, marcado en amarillo.

3. Esperar que cargue todo el programa, alcanzando el 100%.

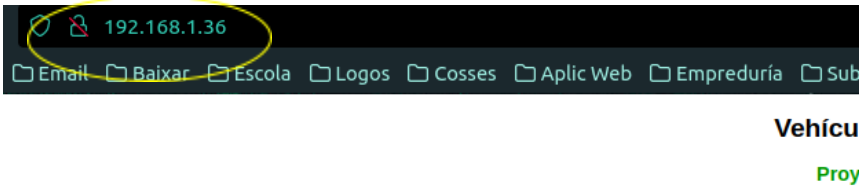


4. A continuación, pulsar en Herramientas y monitor serie.
5.

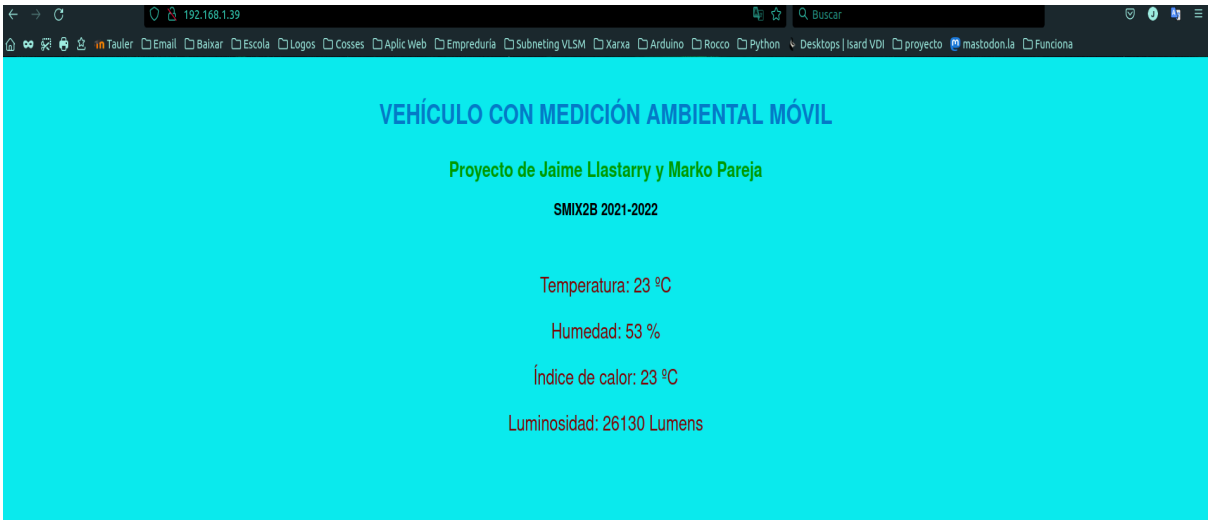


Como observamos en la imagen, se abre un terminal que nos indica el dato que nos interesa, la **ip address** que es el dato que pondremos en nuestro buscador URL, para poder ver las mediciones en tiempo real.

Para entendernos mejor:



6. Y como resultado final, observamos la web que hay preparada con este kit.



Enhorabuena. Si ha llegado a este punto, le felicitamos muy gratamente, ya que no es un kit fácil, pero tampoco muy difícil. Esperamos que haya disfrutado con este kit de Medición. Le invitamos a seguir probando diferentes kits.

Preguntas frecuentes.

1. Placa Nodemcu.
    a. ¿Qué hago si la luz de la placa no se enciende?
        i. No está conectada a la alimentación, o las baterías están descargadas.

    Solución: Conectarla a la alimentación eléctrica (USB ordenador) o reponer las baterías descargadas o defectuosas.

        ii. La placa está o tiene fundido algún componente por sobretensión.

    Solución: Reponer lo antes posible la placa o reparar lo fundido.

        iii. Revisar que el programa está en marcha.

    Solución: Subir de nuevo el programa Arduino, asegurándonos que es el correcto para ver el LED encendido.

    b. Si pulsas cualquiera de los dos botones, reset y flash, y no hacen nada, ¿qué hago?, ¿se ha estropeado la placa?
        i. Los botones mandan una orden a la placa.

    Solución: Esperar 15 segundos para que vuelva a su estado de trabajo. Si pasado este tiempo y no se enciende el LED, la placa está defectuosa: reponerla. Si pasado este tiempo el LED se enciende: Funciona bien, hay que esperar hasta que se inicie de nuevo el programa.

        ii. Los botones no hacen clic ni nada parecido.

    Solución: Los botones están defectuosos: Reponer placa.

2. Protoboard.
    a. No funcionan las conexiones.

    Solución: Los cables machos han de estar bien introducidos. Retirarlos y volver a insertarlos en su lugar. Si el problema persiste, reemplace los cables.

    b. Suena algo suelto dentro.

    Solución: Reemplace inmediatamente la protoboard.

3. Cableado.
    a. Cuando están conectados, no funcionan los elementos.
        i. No hacen contacto.

    Solución: Revisar con firmeza que los bornes del cable, hagan bien contacto con los componentes o elementos.

    Revisar que el cable “toque” o haga contacto con los bornes del mismo. Sustituir por otros cables.

4. Sensores.
    a. DHT11 / El sensor tiene una bombilla y cuando está conectado a la protoboard, no se enciende.

Solución: ¿El cableado está bien conectado? Revisar.
¿Tiene el programa Arduino en marcha? Poner en marcha.
¿Está el kit alimentado eléctricamente? Revisar.
Si todo lo anterior está bien, y no funciona, sustituir el sensor.

    b. GY-30 / El sensor no emite ninguna luz.

Solución: No posee ningún LED con el que ver su funcionamiento.

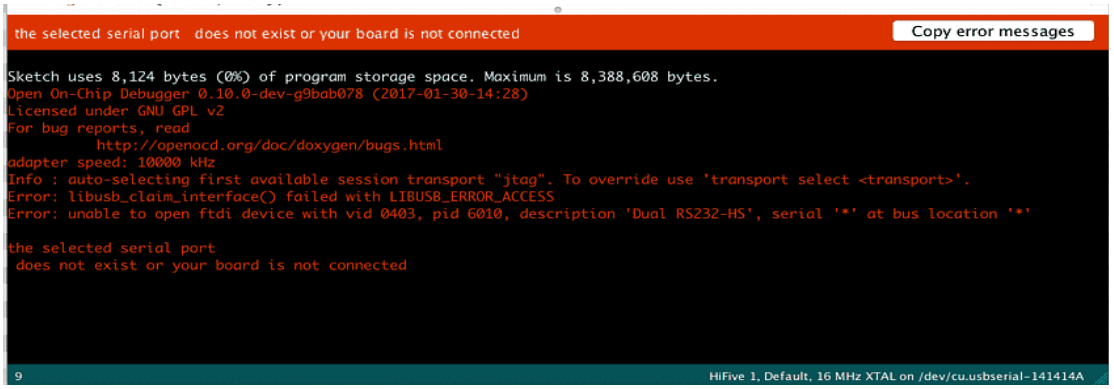
    c. DHT11/GY-30 / Si están conectados al resto de componentes, el programa en marcha y no se ve en pantalla las mediciones, ¿como lo puedo solucionar?

Solución: Revisar paso a paso, los detalles del Proceso de Montaje de este manual.

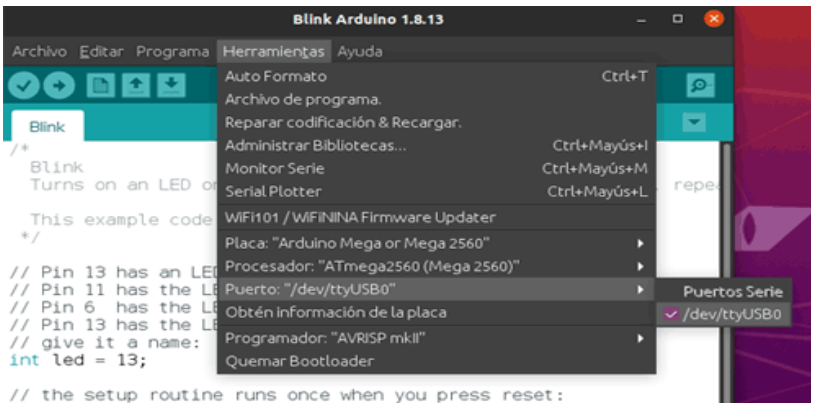
5. Instalación de los programas.
    a. He seguido las instrucciones proporcionadas y no coincide lo que veo en pantalla con los pasos del manual.

Solución: Posiblemente, haya instalado una versión más reciente de los programas, pero no se preocupe, el programa funcionará perfectamente.

    b. Al hacer clic en subir, sale un error como este:



Solución: Ésto sucede, porque normalmente, no nos acordamos de activar el puerto de conexión con Arduino. Diríjase al menú Herramientas, baje un poco hasta visualizar la línea “Puerto: “/dev/ttyUSB”, y asegúrese que está seleccionado el puerto de la conexión con el Arduino. Acto seguido, todo funcionará bien.



c. Falla la conexión wifi con la placa Nodemcu.

Solución: Asegúrese de haber indicado, en el programa Arduino, los datos de la red wifi de su hogar y la contraseña, de no hacerlo, no podrá utilizar dicha placa. Se modifica en estas líneas:

```
// Añadir los datos de vuestra red

const char* ssid = "[redacted]";
// const es como una variable pero no se debe cambiar su valor
const char* password = "[redacted]";
```

d. En la URL sale un error de conexión.

Solución: Cuando se acaba de subir el programa Arduino, la placa Nodemcu tiene un máximo de dos minutos para conectarse a la wifi seleccionada. Pasado estos dos minutos, actualizar la URL (F5) y usted ya podrá ver la web con los datos en tiempo real.

e. Los datos o mediciones no llegan al ordenador.

Solución: Posiblemente la distancia entre el kit montado y el ordenador receptor de los datos, sea muy lejana. Tenga en cuenta que ésta distancia está establecida entre los 300 y 500 metros en línea recta. O, por otro lado, puede ser que cerca del receptor y/o dónde se encuentre, haya muchas interferencias externas. Cambie de ubicación para solucionar el problema.

f. Sale un error al subir el programa que dice “Placa Nodemcu no conect”.

Solución: Esto se debe al refresco de la propia placa. No se preocupe. Vuelva a subir el programa y funcionará perfectamente.

Gracias por utilizar este manual.

10 Manual de usuario WEB coche robot

PINCHA AQUI PARA VER LA WEB

O COPIA ESTO : https://cochesintesi.000webhostapp.com

11 Licencia Creative Commons.

Atribución-NoComercial 4.0 Internacional (CC BY-NC 4.0)

This is a human-readable summary of (and not a substitute for) the license. Advertencia.

Usted es libre de:

Compartir — copiar y redistribuir el material en cualquier medio o formato

Adaptar — remezclar, transformar y construir a partir del material

La licenciante no puede revocar estas libertades en tanto usted siga los términos de la licencia

Bajo los siguientes términos:

Atribución — Usted debe dar crédito de manera adecuada, brindar un enlace a la licencia, e indicar si se han realizado cambios. Puede hacerlo en cualquier forma razonable, pero no de forma tal que sugiera que usted o su uso tienen el apoyo de la licenciante.

NoComercial — Usted no puede hacer uso del material con propósitos comerciales.

No hay restricciones adicionales — No puede aplicar términos legales ni medidas tecnológicas que restrinjan legalmente a otras a hacer cualquier uso permitido por la licencia.

Avisos:

No tiene que cumplir con la licencia para elementos del materiale en el dominio público o cuando su uso esté permitido por una excepción o limitación aplicable.

No se dan garantías. La licencia podría no darle todos los permisos que necesita para el uso que tenga previsto. Por ejemplo, otros derechos como publicidad, privacidad, o derechos morales pueden limitar la forma en que utilice el material.