



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



Escola Politècnica Superior d'Alcoi
Universitat Politècnica de València

Implementación de sensores geolocalizados y una aplicación para la obtención de datos en un área metropolitana

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: David Rodriguez Martinez

Tutor: David Cuesta Frau

Curso 2015-2016

Resumen

El motivo de este proyecto es utilizar una plataforma de "hardware libre" llamada Arduino con el fin de implementar uno o varios nodos de sensores el cual enviará a través de la red Información sobre un área específica, y podremos monitorizar la información gracias a una aplicación web.

Mi intención es que gracias a lo explicado en el párrafo anterior se pueda implementar una utilidad hardware y una Web para utilizar en una Smart City

Palabras clave: Ciudad Inteligente, Arduino, Hardware, Geolocalizacion, Sensores, Hardware Libre, Electronica, Servicios Web

Resum

El motiu d'aquest projecte es utilitzar una plataforma de "hardware lliure" anomenada arduino amb l'intenció d'implementar u o varios nodes de sensors, el cual enviara utilitzant una xarxa, informació sobre una àrea específica la cual podrém monitoritzar gracies a una aplicació web .

La meua intenció es que gracies a el que he explicat en el parraf anterior es puga implementar una utilitat de gestió per utilitzar en una Smart City

Paraules clau: Ciutat Inteligent, Arduino, Hardware, Geolocalizació, Sensors, Harware Lliure, Electrònica, Servicis Web

Abstract

The purpose of this project is use a special Open-source hardware platform called Arduino, with the intention of implement one or more nodes of sensors, which will send on the network, information about a specific area that we can monitoring, using a web service.

My intention is that thanks to I've explained in the last paragraph, I will can implement a hardware and web application to track that information to use in the future in the Smart city.

Key words: Arduino, Smart City, IoT, Remote Sensor, Open-source hardware

Índice general

Índice general	v
Índice de figuras	vii
Índice de tablas	vii
<hr/>	
1 Introducción	1
1.1 Objetivos	1
2 Estudio del Arte	3
2.1 ¿Porque Arduino?	3
2.2 Presupuesto	4
3 Descripción del proyecto	7
3.1 Tecnologías utilizadas	7
3.1.1 Arduino	7
3.1.2 GPS y Nmea Data	8
3.1.3 Nmea Data	8
3.1.4 Formato del NMEA Data	8
3.1.5 Laravel	9
3.1.6 MySQL	9
3.1.7 Bootstrap	9
3.1.8 Chart.js	9
4 Montaje del Hardware y el Servicio Web	11
4.1 Material	11
4.2 Montaje del Nodo Arduino DUE	14
4.3 Programación de la Microcontroladora Arduino DUE	16
4.4 Instalación del Servicio WEB	20
5 Explicación de la aplicación	23
6 Conclusión y mejoras	25
Bibliografía	27

Índice de figuras

3.1 Logo Arduino	7
3.2 Logo de Laravel 5	9
4.1 Diagrama de montaje del proyecto	11
4.2 Microcontroladora Arduino DUE	12
4.3 Ethernet shield	12
4.4 Sensor de Temperatura y Humedad DHT11	12
4.5 Sensor de Temperatura y Humedad DHT11	13
4.6 Sensor de Mesura de Gas CO MQ-7	13
4.7 Sensor de Luminosidad y Ruido	14
4.8 Esquema del montaje	15
4.9 Esquema del circuito para el GPS en el Arduino UNO	15
4.10 Montaje final del nodo de sensores	16
4.11 String JSON generada por la petición GET de Arduino	20
4.12 Configuración del Servidor MAMP	21

Índice de tablas

2.1 Consumo de los diferentes dispositivos	3
--	---

CAPÍTULO 1

Introducción

En la actualidad cada día aumentan el consumo de productos, la necesidad de demanda, la producción en las fábricas, el aumento del tráfico,... todo esto ha generado en el mundo un gran problema de contaminación.

Con este problema he decidido en crear una plataforma para poder medir este problema de una manera sencilla y barata para que cualquier usuario pueda tener acceso a esta información ideando como una especie de plataforma para la medición de contaminación en puntos específicos y este mandar la información vía protocolo HTTP a un servidor en el que puede acceder cualquiera para compartir sus datos.

Este proyecto puede ser útil ya que la aplicación web realizada podrá servir para el uso de gestión de una *SMART City*, o simplemente para recoger información y realizar estudios de contaminación de ciertos lugares ya que la aplicación podría servir esos datos en ficheros para aplicarlos a otros programas y realizar análisis.

1.1 Objetivos

El objetivo de este proyecto es trabajar una parte de informática electrónica y comunicar esta parte con un servidor para poder trabajar esta información y poder servirla a un cliente a través de una aplicación web la cual el usuario podrá obtener y visualizar esa información.

Los objetivos son los siguientes:

- Desarrollo de la plataforma Hardware, en este caso un Arduino Due.
- Configuración de la plataforma para que pueda enviar la información al servidor vía HTTP.
- Configurar el servidor para que ofrezca un servicio web y a la vez reciba la información del Arduino.
- Configurar la base de datos para que almacene la información recibida por la plataforma Hardware.
- Crear la aplicación web que sirva esta información detallada.
- Esta aplicación ademas ofrecerá estas funciones:
 - Podrá servir información en formato .txt o .csv para poder trabajar con ellos.

CAPÍTULO 2

Estudio del Arte

En este capitulo se hablará de porque se han elegido las tecnologías que se van a utilizar en el proyecto así como mostrar algunas de las opciones adicionales que se podrían haber utilizado.

2.1 ¿Porque Arduino?

Arduino es una plataforma Hardware Open Source lo que implica que todo el sistema esta disponible a los usuarios y esto ayudara al proceso de montaje ya que una gran parte de personas (la comunidad de Arduino) trabajan de manera desinteresada en el desarrollo de librerías y aplicaciones para esta plataforma.

Sin embargo Arduino dispone de numerosas versiones por lo que dentro de las mas famosas que hay he decidido analizar cual de los que hay me ofrece mas garantías a la hora de realizar el proyecto por lo que en principio me decantare por el consumo de las microcontroladora, puertos y conexiones disponibles y disponibilidad.

En primer lugar he realizado un estudio del consumo de las placas disponibles para poder realizar este proyecto,

Board	Consumo(W/h)	Consumo(mA/h)
Arduino UNO	0.23	46
Arduino DUE	0.375	75
Arduino MEGA	0.465	93
Arduino Nano	0.075	15
Raspberry Pi	1.77	353

Tabla 2.1: Consumo de los diferentes dispositivos

En este caso nos interesaría el elegir el dispositivo que menor consumo tenga pero también he tenido que mirar que funcionalidad pueden aportar cada placa, cabe destacar que la Raspberry Pi no es una microcontroladora, si no un ordenador de placa reducida, pero ofrece una funcionalidad que podría mejorar las prestaciones de este proyecto, hablando de seguridad y comunicación, pero sin embargo después de las pruebas de consumo comparando con el Arduino que mas consume, en este caso el MEGA, consume en mA un 379 % mas ya que necesita mantener un sistema operativo que aunque sigue siendo un consumo muy bajo para ser un ordenador, es muy elevado para este proyecto así que en este punto descartamos la Raspberry Pi.

Otro punto a tener en cuenta son las capacidades de conexión que tienen las microcontroladoras, ya que para poder conectarlo todo voy a necesitar 3 entradas analógicas, una entrada digital, 2 entradas para el rx y tx del puerto serie y compatibilidad con la Shield ethernet de Arduino.

El Arduino UNO es una muy buena opción ya que tiene todo esto y es de un consumo reducido aun así, la comunicación serial va por software ya que hay que implementar una librería especial para poder realizar un puerto serie para comunicarse con el GPS, ademas habría que diseñar un circuito por los problemas en los voltajes de las lecturas digitales ya que algunos componentes devuelven 3.3V y Arduino los detecta a 5V, que bien que no daría problemas pero podría dar lugar a valores anómalos en determinadas ocasiones.

El Arduino MEGA es la mejor opción disponible sin embargo ajustando con el material disponible, he de elegir el Arduino DUE ya que tiene un consumo mas reducido.

Estas son las características del Arduino DUE:

- **Microcontrolador:** AT91SAM3X8E
- **Velocidad de reloj:** 84 MHz
- **Tensión de trabajo:** 3.3V
- **Pines de entradas digitales:** 54
- **Pines de entradas analógicas:** 12
- **Memoria Flash:** 512 KB

He subrayado los valores que son importantes ya que en el estudio del montaje en el Arduino UNO tenia problemas con la memoria del programa ya que la memoria Flash quedaba casi completa utilizando la librería que implementaba un serial en cualquier pin digital, imposibilitando la capacidad de poder implementar mejoras en la aplicación.

2.2 Presupuesto

En esta parte del proyecto se va a realizar un estudio de lo que puede llegar a costar la creación, el montaje y el mantenimiento de la aplicación, todo ello al precio actual del año 2016.

Em primer lugar se expondrá el coste de que costaría aproximadamente cada nodo de la aplicación:

- **Arduino DUE:** 36€
- **Shield de Ethernet:** 26€
- **Neo6Mv2:** 13,50€
- **Sensor de Gas MQ-7:** 8€
- **Sensor de Temperatura y Humedad DHT11:** 3€
- **Sensor de Sonido:** 0,50€
- **Sensor de Luz:** 1€

Todo esto nos da un coste de **88 €** por nodo de sensores aproximadamente.

Se podrían utilizar materiales mas baratos con el fin de reducir los costes aun mas pero no seria aconsejable pues del proyecto interesa que este activo el mayor tiempo posible. También se pueden modificar los elementos del mismo, nombrados en la parte donde se realizaba el estudio del consumo de energía de cada controladora (**Tabla 2.1**) realizando las modificaciones necesarias para que este pueda funcionar.

También se ha incluido el coste de lo que costaría añadir el servidor de la aplicación, en este caso he puesto componentes genéricos:

- **Servidor:** 600€
- **Router:** 30€
- **Cableado:** 12€

Todo esto nos da un coste de **642 €** por servidor aproximadamente, pero, como se ha explicado antes, todo esto se puede reducir con material ya disponible, porque, "quien no tiene un ordenador hoy día?".

Y por ultimo el mantenimiento de lo que seria todo el montaje, ya es difícil hablar de un mantenimiento pues la aplicación pues aumentaría con cada nodo añadido, la ampliación del servidor en caso de haber demasiadas peticiones, el modo de alimentar los nodos que podría ser por microUSB, POE o baterías, diferentes Controladoras, todo esto nos daría un resultado muy variable, así que es muy complicado cuanto costaría mantener la aplicación.

CAPÍTULO 3

Descripción del proyecto

Para este proyecto voy a utilizar una micro controladora Arduino DUE desatollada por **Arduino**, en esta se implementaran una serie de sensores y librerías que permitirán realizar una serie de funciones. Para enviar la información a la red, dicha controladora necesitara una Shield que le permita conectarse a la red también desatollada por **Arduino**.

Para recoger toda esta información se diseñara una aplicación que reciba las peticiones y pueda almacenarlas en una base de datos, a su vez esta aplicación web también servirá esta información al usuario de una manera detallada para su estudio.

Finalmente se le ofrece al usuario generar unos reportes en los que este podrá descargarlos para, en un futuro, poder trabajar con esa información en otros programas o plataformas.

Esta es la explicación de las tecnologías y dispositivos que voy a utilizar:

3.1 Tecnologías utilizadas

3.1.1. Arduino

Arduino DUE es una placa Microcontroladora basada en el Atmel SAM3X8E ARM Cortex-M3 CPU. Es el primer Arduino basado en un Núcleo microcontrolador ARM de 32 bits ya que su versión anterior (Arduino UNO), trabaja directamente sobre un microcontrolador.



Figura 3.1: Logo Arduino

Para programar estas controladoras es necesario utilizar el IDE de Arduino, realizando una comunicación serie entre la maquina y la controladora podremos realizar un flash de la memoria con el fin de incorporar a la memoria interna del Arduino DUE el programa que hemos diseñado para realizar las funciones. Para programarlo se utiliza una versión simplificada de C++ la cual realiza una configuración inicial (Setup) y un bucle infinito (Loop), una vez arrancado el programa en la controladora este se repetirá indefinidamente realizando siempre la función programada.

El IDE de Arduino podemos descargarlo de aquí:

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

3.1.2. GPS y Nmea Data

Para el proyecto vamos a utilizar el GPS Neo6mv2, este GPS es bastante útil ya que es barato y de bajo consumo, ideal para la propuesta del proyecto ya que lo que se interesa es el ahorro de energía.

Este GPS nos podrá enviar información útil del satélite como la posición, la altitud, la fecha y la hora todo utilizando un formato de cadena Hexadecimal llamada NMEA DATA

3.1.3. Nmea Data

"El National Marine Electronics Association (NMEA) ha desarrollado una especificación que define la interfaz entre varias piezas de equipos electrónicos. La norma permite la electrónica de la marina enviar información a los ordenadores ya otros equipos marinos."

Los receptores GPS están incluidos en esta especificación. Muchos de los programas de Posicionamiento están comprendidos bajo el formato NMEA. Estos datos incluyen PVT (Posición, Velocidad, Tiempo) generada por el receptor GPS. La idea del NMEA es enviar información llamada frase la cual es totalmente independiente de otras frases.

3.1.4. Formato del NMEA Data

El modulo Neo6Mv2 funciona enviado al serie cadenas de números hexadecimales en formato de NMEA data. El GPS envía información al serie cada intervalo de tiempo por lo que no sería necesario el puerto Tx del Arduino ya que solo se va a recibir información, no obstante, lo he configurado con el fin de poder configurar lo desde el Arduino.

Este es el resultado que envía el GPS por el Serial2 configurado en Arduino:

```

1 $GPRMC,192128.00,V,,,,,,150216,,,N*7D
2 $GPVTG,,,,,,N*30
3 $GPGGA,192128.00,,,,,0,00,99.99,,,,,*67
4 $GPGSA,A,1,,,,,,,,,,99.99,99.99,99.99*30
5 $GPGSV,1,1,01,06,,,18*77
6 $GPGLL,,,,,192128.00,V,N*4B

```

Si analizamos los datos, siguiendo la especificación de la NMEA data se observa que recibe información del GPS pero analizando en concreto las dos cadenas \$GPGGA se puede apreciar que no se recibe la información de la posición por lo que hay que posicionar el modulo GPS directamente en contacto con el satélite, una vez así este establecerá conexión y enviará esta cadena:

```

1 $GPRMC,192232.00,V,,,,,,150216,,,N*75
2 $GPVTG,,,,,,N*30
3 $GPGGA,192232.00,,,,,0,00,99.99,,,,,*6F
4 $GPGSA,A,1,,,,,,,,,,99.99,99.99,99.99*30
5 $GPGSV,2,1,06,02,35,106,52,06,28,058,48,12,69,064,46,19,11,045,49*7D
6 $GPGSV,2,2,06,21,,,35,24,40,151,4

```

3.1.5. Laravel

Basado en *Symphony* Laravel es un framework Opensource que permite desarrollar aplicaciones y servicios web sirviéndose de una estructura Modelo-Vista-Controlador (MVC) ya creada en PHP 5.

Con este entorno desarrollare la aplicación web que nos servirá para almacenar los datos y usara algunas tecnologías para ello.



Figura 3.2: Logo de Laravel 5

3.1.6. MySQL

MySQL es un SGBD (Sistema de Gestión de Bases de Datos) relacional bajo licencia dual GPL por Oracle está considerada como la base datos open source más popular para entornos de desarrollo de aplicaciones Web.

3.1.7. Bootstrap

Bootstrap es una librería CSS para el desarrollo de vistas para aplicaciones web. Utilizado para desarrollar la interfaz de usuario en paginas web, como los botones, formularios, cabeceras...

3.1.8. Chart.js

Chart.js es una librería JavaScript, utilizada para generar gráficos en el la vista de las aplicaciones web, es capaz de generar gráficos dinámicos, útil para el proyecto pues se implementara una serie de gráficos que monitorizaran la información obtenida.

CAPÍTULO 4

Montaje del Hardware y el Servicio Web

En este capítulo se va a explicar como se ha montado todo este sistema, así como una explicación de sus diferentes piezas y la función que realiza cada una de ellas tanto en la parte hardware como en la parte software. También se realizará un estudio del coste de todos los materiales, así como el mantenimiento y consumo de estos.

Esta es la estructura que se seguirá para el montaje de la aplicación:

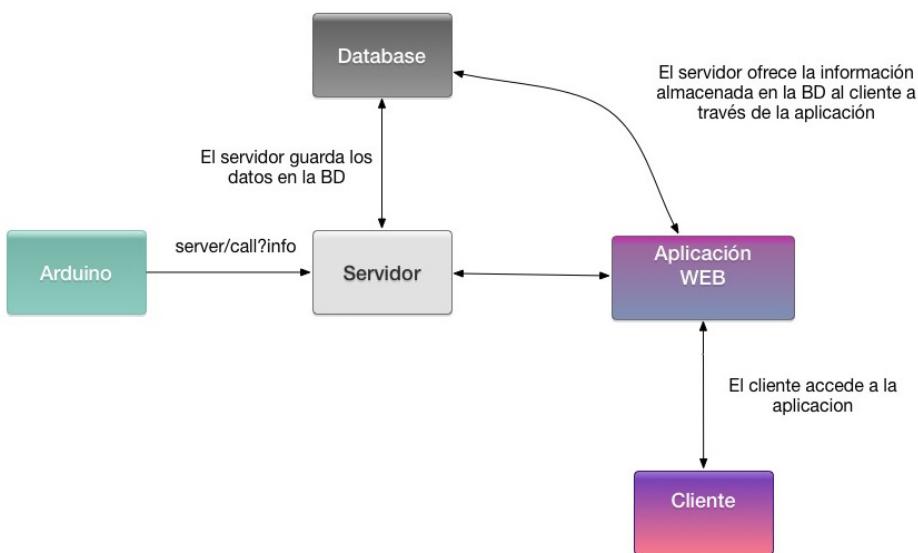


Figura 4.1: Diagrama de montaje del proyecto

En la **Figura 4.1** se puede observar el diagrama de montaje del proyecto en que función realizan las diferentes partes, desde el módulo hasta la información que podrá visualizar el cliente.

4.1 Material

Para este proyecto se ha utilizado el siguiente material:

Arduino Due: Es la Controladora que se encarga de recibir la información y procesarla para enviarla al servidor.

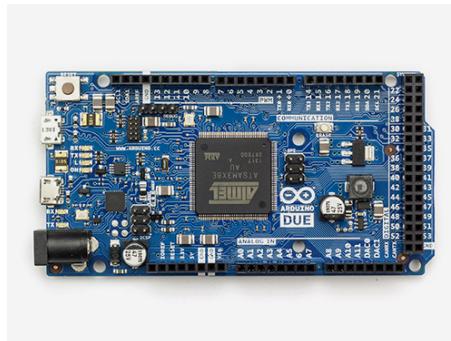


Figura 4.2: Microcontroladora Arduino DUE

Ethernet shield: Es una Shield que permite a la micro controladora conectarse a la red.

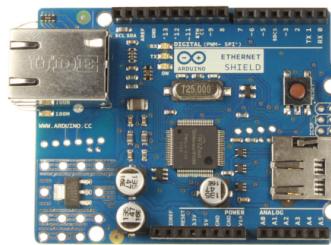


Figura 4.3: Ethernet shield

Modulo GPS NEO6mv2: Es un módulo el cual contiene un GPS que una vez este conectado al satélite enviara información de la geolocalización.



Figura 4.4: Sensor de Temperatura y Humedad DHT11

Sensores: Un conjunto de sensores que permitirá la obtención de los valores ambientales del entorno donde este instalado el nodo.

Estos sensores son:

- **DHT11** Es el sensor de Humedad y temperatura de Adafruit recogerá estos datos climáticos del ambiente.

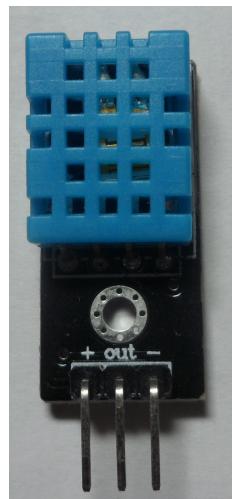


Figura 4.5: Sensor de Temperatura y Humedad DHT11

- **Sensor de gas CO MQ-7 y el chipset LM393** el que se utilizara para la medición de gases en el ambiente.



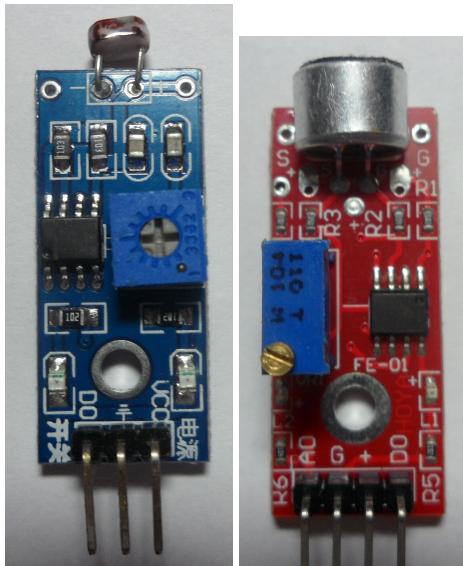
Figura 4.6: Sensor de Mesura de Gas CO MQ-7

- **Sensor luz y sonido** el cual también utiliza el chipset LM393 medirá la contaminación lumínica y acústica.

Servidor: El servidor que se va a utilizar para realizar las función de conexión entre el nodo y la la base de datos y entre el cliente y la conexión con el cliente a través de la aplicación web.

Las características de este servidor serán:

- **CPU:** Intel i5 2500K a 3.3 GHz de funcionamiento.
- **Memoria:** 8 GB 1600 MHz DDR3.
- **Almacenamiento:** 2 TB de Almacenamiento, el que permitirá tener una base de datos bastante amplia.
- **S.O:** OSX El Capitan 10.11.6.



(a) Sensor de Lumino-
sidad (b) Sensor de Ruido

Figura 4.7: Sensor de Luminosidad y Ruido

Aplicaciones: Aplicaciones que me han permitido realizar el montaje de todo el sistema. Entre ellas están:

- **Mamp:** Es una aplicación para montar un servidor web Apache y Mysql bajo los puertos 8888 para el web y 8889 para el Mysql.
- **MySQL Workbench** aunque es cierto que Laravel 5 gestiona la base de datos través de PHP se ha utilizado esta aplicación para poder gestionar la información almacenada en la base de datos.
- **PhpStorm** Un IDE de desarrollo para programar con PHP así como gestionar proyectos con Git.

Cableado: Distintos tipos de cableado para conectarlo todo.

4.2 Montaje del Nodo Arduino DUE

El nodo para esta aplicación sera la microcontroladora con los módulos de sensores que enviaran información al servidor, para ello se realizara el montaje siguiendo el siguiente esquema:

En la **Figura 4.8** se puede apreciar el esquema de las conexiones que he seguido para el montaje del nodo para que el código de la aplicación para Arduino recoja los valores correctamente.

Como se puede apreciar en el esquema el montaje es sencillo ya que no necesita electrónica adicional como resistencias u otros componentes electrónicos. Para conectar esto cada uno de los sensores, siguiendo el esquema, se conectara VCC (**Rojo**) a 3.3V ya que es el funcionamiento de cada uno de los sensores exceptuando el GPS que ira conectado a 5V ya que ha demostrado un mejor funcionamiento a ese voltaje y GND (**Negro**) para el negativo de los módulos, todos los sensores exceptuando el sensor de temperatura y

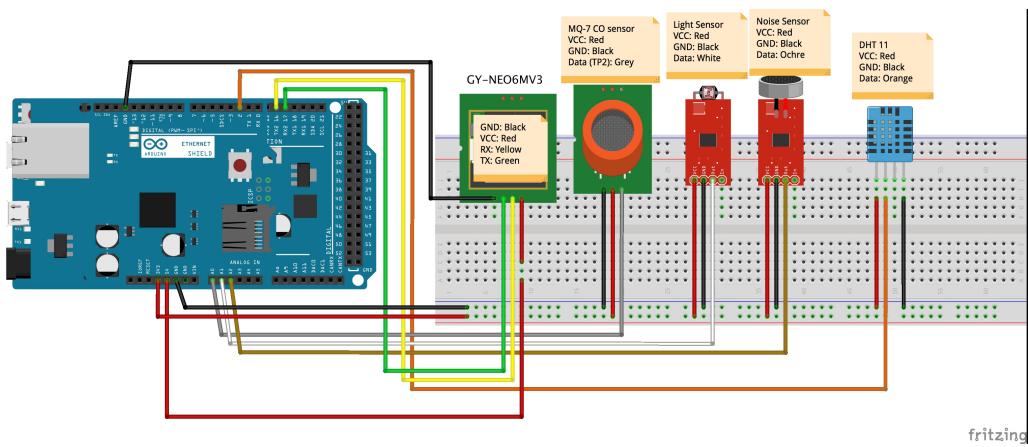


Figura 4.8: Esquema del montaje

humedad DHT11 que va a la entrada digital y el GPS que va conectado al Serial 2, van conectados a entradas analógicas (de A0 a A2).

Tener en cuenta que si se realiza en un Arduino UNO al ser una versión mas antigua las lecturas digitales se leen en un rango de 5V por lo que hay que diseñar un circuito especial para poder leer el GPS y el sensor DHT11 ya que estos devuelven una señal digital de 3.3V, para solucionar esto se tendrían que poner resistencias en el circuito con el fin de diseñar un sistema que aumente la señal a los 5V que necesita el Arduino UNO, que en este caso quedaría tal que así:

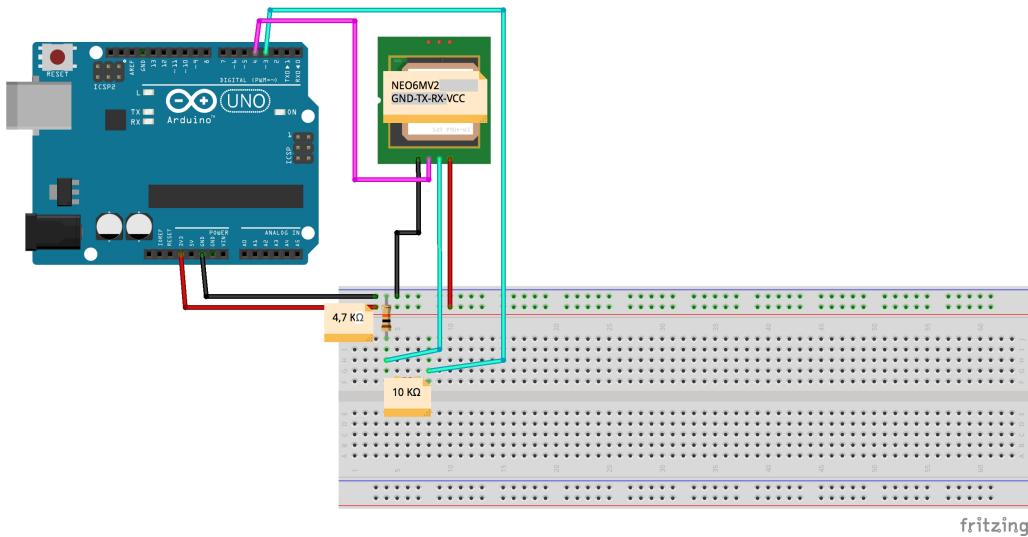


Figura 4.9: Esquema del circuito para el GPS en el Arduino UNO

El circuito de la Figura 4.9 es un ejemplo de instalación para un modulo en Arduino UNO cuya respuesta en digital sea de 3.3V y sirve para otros módulos.

También comentar que los RX y TX de la trasmisión serie van conectados de manera invertida entre el GPS y el Arduino.

Este seria el montaje final de la electrónica de la controladora con todo el cableado y funcionando enviando la información al servidor.

Cada uno de estos sensores tendrá su función en el código de manera modular, así que es mas sencillo entender el funcionamiento del mismo.

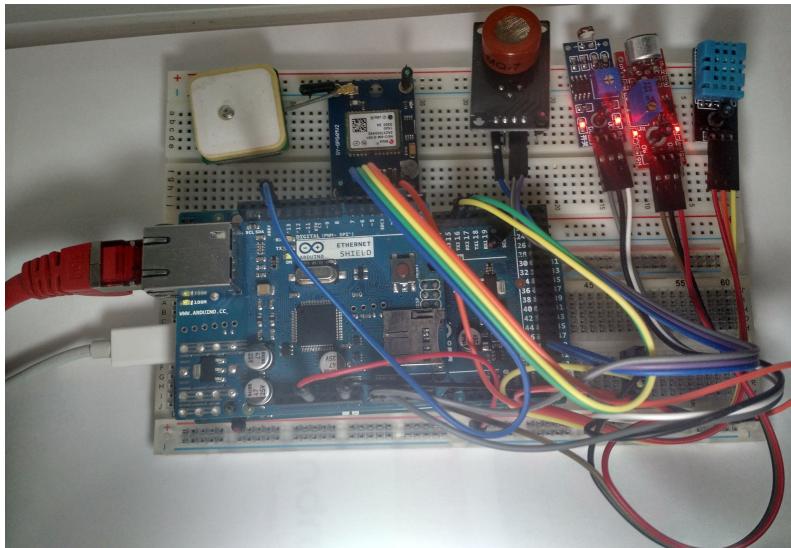


Figura 4.10: Montaje final del nodo de sensores

4.3 Programación de la Microcontroladora Arduino DUE

En este apartado se va a explicar cada una de las partes del código que utilizará el Arduino DUE, dicha parte realizará una función en la controladora por lo que el conjunto de todo el código realizará la función propuesta para este proyecto.

El código se puede descargar de la siguiente URL: https://www.dropbox.com/s/qzzff1y6fqgluh8/project_1.3.1.ino?dl=0

Esta es la parte de la configuración:

```

1 /*
2 * FILE:      project-v1.3.1.ino
3 * AUTHOR:    David Rodriguez Martinez davidrm146@gmail.com
4 * VERSION:   1.3.1
5 */
6 //Necessary Libraries
7 #include <Ethernet.h>
8 #include <SPI.h>
9 #include <TinyGPS.h>
10 #include "DHT.h"
11
12 //Config of the device
13 byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
14 TinyGPS gps;
15 #define DHTPIN 2
16 #define DHITYPE DHT11
17 DHT dht(DHTPIN, DHITYPE);
18 IPAddress ip(192, 168, 1, 102);
19 EthernetClient client;
20 String data;
21 void setup()
22 {
23     Serial.println("EXECUTING setup");
24     Serial.begin(9600);
25     Serial2.begin(9600);
26     if (Ethernet.begin(mac) == 0) {
27         Serial.println("Failed to configure Ethernet using DHCP");
28         Ethernet.begin(mac, ip);
29     }

```

```

30 } delay(1000); // A Small delay for avoid errors
31 }
```

Antes de entrar al código se implementaran las librerías para el funcionamiento de los módulos, en este caso lo que hacen es traducir la información para que pueda ser manipulada de manera mas fácil. las librerías son:

- #include <Ethernet.h>: Permite gestionar la conexión a la red.
- #include <SPI.h>: Permite al Arduino DUE comunicarse con la Shield de Ethernet.
- #include <TinyGPS.h> Esta librería traduce el NMEA data en valores legibles sin necesidad de manipular la cadena de hexadecimales recibida.
- #include 'DHT.h': Esta librería traduce la información recibida por el sensor DHT11, solo hace falta configurar que pin es el que recibirá la información y que tipo de sensor se va a utilizar ya que la librería configura automáticamente la comunicación con el sensor.

Esta es la configuración de Arduino encapsulada en la definición de SETUP en esta parte del código se definirá la comunicación vía serie para el modulo GPS conectado al serial 2 a 9600 baudios aunque aplicando una serie de cadenas hexadecimales en el arranque del programa escritas en el GPS vía serie. También se configurara la conexión serie que le pertenece al puerto USB para poder observar la información que envía cada vez que se repita el bucle del programa con el fin de poder debuggearlo en caso de algún problema o fallo de los componentes.

```

1 if (Ethernet.begin(mac) == 0) {
2     Serial.println("Failed to configure Ethernet using DHCP");
3     Ethernet.begin(mac, ip);
4 }
```

Esta parte es la conexión a la red, en este caso por DHCP que solo hace falta asignarle la MAC al dispositivo o si falla asignar manualmente una dirección IP que corresponderá a la de la red interna.

Una vez configurado todo el programa ya estará listo para leer la información que recojan los módulos de sensores asi que se dieñara el siguiente codigo para realizar dicha lectura en la definición del LOOP el cual se ejecutara indefinidamente:

```

1 void loop(){
2     Serial.println("EXECUTING LOOP");
3 //*****GPS CODE*****
4     bool newData = false;
5     unsigned long chars;
6     unsigned short sentences, failed;
7     for (unsigned long start = millis(); millis() - start < 1000;){
8         while (Serial2.available()){
9             char c = Serial2.read();
10            if (gps.encode(c))newData = true;
11        }
12    }
13    float flat, flon;
14    unsigned long age;
15    gps.f_get_position(&flat, &flon, &age);
16    Serial.print("LAT=");
17    Serial.print(flat == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flat, 6);
```

```

18 Serial.print(" LON=");
19 Serial.print(flon == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : flon, 6);
20 Serial.print(" SAT=");
21 Serial.print(gps.satellites() == TinyGPS::GPS_INVALID_SATELLITES ? 0 : gps.
    satellites());
22 Serial.print(" PREC=");
23 Serial.println(gps.hdop() == TinyGPS::GPS_INVALID_HDOP ? 0 : gps.hdop());
24
25 //The next variable keeps the information collected by the sensors
26 data =
27     "board_id=" + getID() +
28     "&temp=" + getTemperature() +
29     "&hum=" + getHumidity() +
30     "&gas=" + getGas() +
31     "&noise=" + getNoise() +
32     "&luz=" + getLight() +
33     "&poslat=" + String(flat, 6) +
34     "&poslon=" + String(flon, 6);
35
36 //Debug function, this string works
37 //data ="board_id=1&temp=23&hum=43&gas=53&noise=54&luz=55&poslat=63&poslon=73";
38 displayData(data);
39 //This function is for send information to the server
40 sendDataGet(data);
41 delay(60000);
42 }

```

En esta parte del código se realizan 3 funciones, la lectura del GPS, el envío de la cadena que contiene las funciones que recogen de datos de los sensores y el delay que establecer un periodo de envío de información con el fin de evitar saturar la red.

```

1 for (unsigned long start = millis(); millis() - start < 1000;){
2     while (Serial2.available()){
3         char c = Serial2.read();
4         if (gps.encode(c)) newData = true;
5     }
6 }

```

Esto recogerá la información del gps en formato de NMEA data pero gracias a la librería TinyGPS.h que nos traduce directamente sin tener que editar manualmente el NMEA data, la información obtenida del GPS leyendo el Serial2 del Arduino DUE devolviendo nos información del GPS, en este caso nos interesa **flat** y **flon** que son las coordenadas que recoge el GPS.

Los Serial.print están para mostrar la información obtenida ya que este no conecta al instante al satélite como ya se vera en el funcionamiento del vídeo mas adelante.

También se recoge la información para enviar al servidor en la variable **data** en la que se llaman las funciones que recogerán la información, es importante que el formato de la cadena sea como esta especificado en los comentarios, siguiendo el nombre de las variables ya que si no están bien nombrados la función del servidor que se encarga de recoger la información no funcionara y no se podrá realizar la inserción a la base de datos. Este es un ejemplo de como debería ser definida:

```

1 data="board_id={id} +
2     &temp={temperatura} +
3     &hum={humedad} +
4     &gas={medicion del gas} +
5     &noise={ruido ambiental} +
6     &luz={luminosidad} +

```

```

7  &poslat={latitud} +
8  &poslon={longitud}";

```

Y por ultimo una delay o pausa que evitara que el Arduino esté enviando información constantemente de 1 minuto (60000 milisegundos)

Una vez explicado el bucle se explicaran las funciones que se ejecutan en la variable de datos ya que son las que recogen la información ambiental de la zona donde esta situado:

```

1 String getTemperature() {
2   float t = dht.readTemperature();
3   float f = dht.readTemperature(true);
4   if (isnan(t) || isnan(f)) {
5     Serial.println("Failed to read Temperature from DHT. . .");
6     return "0";
7   }
8   return String(t, 2);
9 }
10
11 String getHumidity() {
12   float h = dht.readHumidity();
13   if (isnan(h)) {
14     Serial.println("Failed to read Humidity from DHT. . .");
15     return "0";
16   }
17   return String(h, 2);
18 }
19
20 String getGas() {return String(analogRead(A0));}
21
22 String getLight() {return String(analogRead(A1));}
23
24 String getNoise() {return String(analogRead(A2));}
25
26 String getID() {return "1";}

```

En las funciones *getTemperature* y *getHumidity* se puede apreciar que gracias a la librería dht.h se realiza la conexión y lectura de la información del sensor sin tener que crear código adicional para controlar este sensor. Sobre las demás funciones son lecturas analógicas de las mismas que nos dan unos valores normales en su rango de 1 a 1024. Y la ultima sirve para establecer la id en la que esta registrada en la base de datos, este numero debe coincidir con su controladora como ya se explicará mas adelante.

Una vez ya esta explicado el código se explicara la llamada a la web la cual realizara la inserción a la bd, este es su código:

```

1 void sendDataGet(String data) {
2   if (client.connect("192.168.1.16", 8888)) {
3     client.print("GET /call?");
4     client.println(data);
5     client.println("HTTP/1.1");
6     client.println("Host: 192.168.1.16");
7     client.println("Connection: close");
8     client.println();
9   }
10  if (client.connected()) {
11    client.stop();
12  }
13 }

```

Esta función realiza la petición GET al servidor web el cual se configurara con la ip y el puerto en el que esté trabajando el servidor, puede ser publica o privada por lo que el

Arduino puede trabajar en Internet aunque es mucho mas inseguro para la aplicación pues uno de sus puntos débiles es que el Arduino no tiene capacidad para la encriptación y tampoco tiene soporte para peticiones HTTPS por lo que una gran falla de este proyecto son los ataques *man in the middle*, la llamada corresponde a una petición a la web `?call?información....en` la cual se obtendrá la información en forma de JSON y sera almacenada den la BD.

La llamada o dirección del servidor web quedaría algo tal que así:

```
1 192.168.1.16:8888 / call?board_id={id}&temp={temperatura}&hum={humedad}&gas={  
medicion del gas}&noise={ruido ambiental}&luz={luminosidad}&poslat={  
latitud}&poslon={longitud}
```

Este es el código de la pagina web PHP que recogerá los datos en formato JSON y los guardara en la base de datos, todo ello realizado en Laravel 5.

```
1 public function setData(){  
2     $getData = Request::all();  
3     $data = create($getData);  
4     if (!$this->active) {$this->setWorking(Request::get('board_id'))};  
5     return $getData;  
6 }
```

La función `$getData = Request::all();` recoge toda la información que reciba en un GET o POST y lo introduce en la variable en formato JSON luego se ejecutaría el controlador de data la función `data::create($getData)`; de ahí la importancia que las variables introducidas en la variable data de Arduino estén nombradas como se ha especificado ya que en la BD se han de nombrar igual para que las entradas puedan ser introducidas correctamente.

Este es el resultado del JSON cuando se recibe la petición GET en la función:

```
{"board_id": "1", "temp": "26", "hum": "31", "gas": "93", "noise": "35", "luz": "35", "poslat": "38.960045", "poslon": "-0.181004"}
```

Figura 4.11: String JSON generada por la petición GET de Arduino

Una vez configurado esto ya se podría dejar la microcontroladora trabajando para la recolección de datos.

4.4 Instalación del Servicio WEB

Para la instalación en este caso se requiere de un Servidor WEB con la funcionalidad de PHP5 y un SGBD (en este caso MySQL) añadido, en este proyecto se ha utilizado el servidor MAMP que es una aplicación de OSX que dispone de estas características el cual se va a utilizar para almacenar la información.

Para la instalación de la aplicación en el servidor se ha de descargar de este enlace el cual contiene un paquete de archivos que contienen la aplicación:

<https://www.dropbox.com/sh/nf8qp6jtqgrqb9d/AACmTXwt8dyTvCTKeraueG9oa?dl=0>

Una vez descomprimido se le aplican los permisos lectura y escritura al usuario y lectura a otros (`sudo chmod 744 tfg-laravel`) desde la terminal y se configura el directorio del servidor WEB:

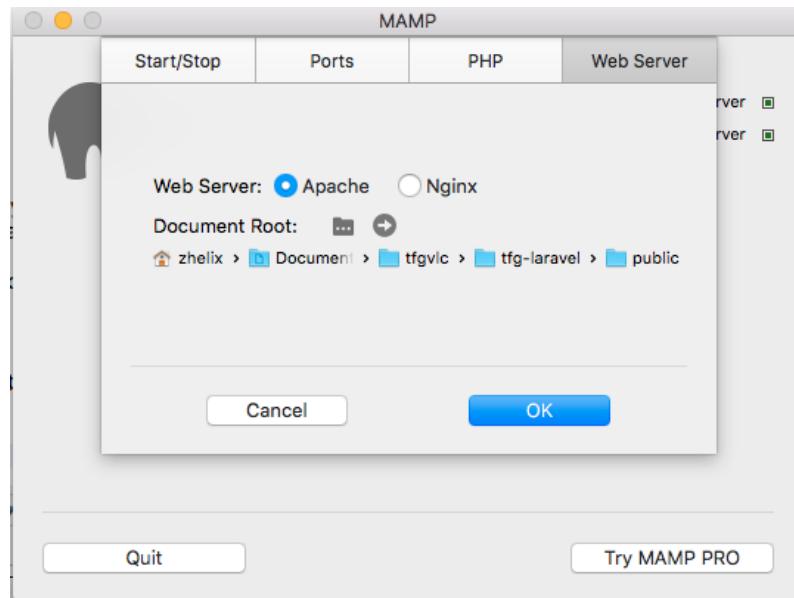


Figura 4.12: Configuración del Servidor MAMP

En la ventana de la **Figura 4.12** se muestra la configuración básica del directorio de la aplicación, que en este caso ha sido alojado en la carpeta de usuario, los demás parámetros se dejaran por defecto como es el puerto HTTP que lo configurara al 8888 y el servidor de Base de Datos que sera el 8889.

CAPÍTULO 5

Explicación de la aplicación

CAPÍTULO 6

Conclusión y mejoras

Una mejora puede ser la utilizacion de

Bibliografía

- [1] Jose Manuel Ruiz Gutierrez Arduino + Ethernet Shield *Funcionamiento de Arduino*, Versión 1, Enero, 2013. (Consultado el 27-12-2015)
- [2] Información sobre Arduino DUE, conexiones y funcionamiento <https://www.arduino.cc/en/Main/ArduinoBoardDue>.
(Visitado el 10-01-2016)
- [3] Información, decodificación, programación y DataSheet del dispositivo de Localización NEO6mv2 utilizado en el proyecto [https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_\(GPS.G6-HW-09005\).pdf](https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_(GPS.G6-HW-09005).pdf).
<http://www.gpsinformation.org/dale/nmea.htm>.
<http://arduiniana.org/libraries/tinygps/>
(Visitado el 15-01-2016)
- [4] Información del funcionamiento y librerías del sensor DHT11 <http://playground.arduino.cc/Main/DHT11Lib>.
<https://cdn-learn.adafruit.com/downloads/pdf/dht.pdf>.
(Visitado el 20-03-2016)
- [5] Información del funcionamiento y DataSheet del sensor LM393 y MQ-7 Sensor CO consultado en <http://www.ti.com/lit/ds/symlink/lm2903-n.pdf>. <https://www.sparkfun.com/datasheets/Sensors/Biometric/MQ-7.pdf>.
- [6] Información sobre como desarrollar una aplicación WEB utilizando el Framework Laravel 5.2 consultado en <https://laravel.com/docs/5.2>.
- [7] Información y descarga de la aplicación para el servidor web obtenido en <https://www.mamp.info/en/>.

