

**Arduino & RasPberry**

**David Moreno Moreno**

**Fernando Donaire GarcÍa**

**Sorin Gavrila**

**Grupo 6 – TRABAJO EN GRUPO 3**

dESARROLLO TECNOLOGÍAS EMERGENTES

gRAdo SISTEMAS DE LA INFORMACIÓN

**CONTENIDO**

[1. Autores del trabajo, planificación y entrega 3](#_Toc450651997)

[1.1 Autores 3](#_Toc450651998)

[1.2 Planificación 3](#_Toc450651999)

[1.3 Entrega 3](#_Toc450652000)

[2. Requisitos del prototipo a implementar 4](#_Toc450652001)

[2.1 Requisitos funcionales 5](#_Toc450652002)

[2.2 requisitos no funcionales 6](#_Toc450652003)

[3. Criterios de comparación en la implementación 6](#_Toc450652004)

[3.1 Criterio 1: Documentación 6](#_Toc450652005)

[3.2 Criterio 2: Horas empleadas en el desarrollo del sistema 6](#_Toc450652006)

[3.3 Criterio 3: Líneas de código 7](#_Toc450652007)

[3.4 Criterio 4: Peso del programa 7](#_Toc450652008)

[3.5 Criterio 5: Tiempo de arranque de tecnologías 7](#_Toc450652009)

[3.6 Criterio 6: Tiempo verificación configuración Ethernet 7](#_Toc450652010)

[3.7 Criterio 7: Rendimiento en cuanto a funcionamiento interrumpido 7](#_Toc450652011)

[3.8 Criterio 8: Rendimiento en cuanto a funcionamiento ininterrumpido 7](#_Toc450652012)

[3.9 Criterio 9: Recursos necesarios 8](#_Toc450652013)

[3.10 Criterio 10: Precio 8](#_Toc450652014)

[3.11 Criterio 11: Espacio físico 8](#_Toc450652015)

[4. Proyecto de implementación de un prototipo del sistema utilizando Arduino 9](#_Toc450652016)

[4.1 Documentación de diseño 9](#_Toc450652017)

[4.1.1 Diagrama de uso lógico 9](#_Toc450652018)

[4.1.2 Diseño interfaz de usuario 9](#_Toc450652019)

[4.2 Documentación de construcción 9](#_Toc450652020)

[4.2.1 Diagrama de conexiones Arduino 9](#_Toc450652021)

[4.2.2 Diagrama de uso lógico del script de Arduino 11](#_Toc450652022)

[4.2.3 Diagrama de uso lógico de la API 13](#_Toc450652023)

[4.2.4 Diagrama de uso lógico de la API 13](#_Toc450652024)

[4.2.5 Diagrama de uso lógico de la interfaz 15](#_Toc450652025)

[4.3 Documentación de pruebas 18](#_Toc450652026)

[4.4 Documentación de instalación 19](#_Toc450652027)

[4.5 Manual de usuario 21](#_Toc450652028)

[5. Proyecto de implementación de un prototipo del sistema utilizando Raspberry 22](#_Toc450652029)

[5.1 Documentación de diseño 22](#_Toc450652030)

[5.1.1 Diagrama de uso lógico 22](#_Toc450652031)

[5.1.2 Diseño interfaz de usuario 22](#_Toc450652032)

[5.2 Documentación de construcción 22](#_Toc450652033)

[5.2.1 Diagrama de conexiones Raspberry 23](#_Toc450652034)

[5.2.2 Diagrama de uso lógico del script de Raspberry 24](#_Toc450652035)

[5.2.3 Diagrama de uso lógico de la API 25](#_Toc450652036)

[5.2.4 Diagrama de la base de datos 26](#_Toc450652037)

[5.2.5 Diagrama de uso lógico de la Interfaz de Usuario 27](#_Toc450652038)

[5.3 Documentación de pruebas 31](#_Toc450652039)

[5.4 Documentación de instalación 32](#_Toc450652040)

[5.4.1 Instalar el sistema operativo de Raspberry 32](#_Toc450652041)

[5.4.2 Configurar el sistema operativo de Raspberry 34](#_Toc450652042)

[5.4.3 Configurar servidor web 36](#_Toc450652043)

[5.5 Manual de usuario 39](#_Toc450652044)

[6. Comparación de las dos implementaciones 40](#_Toc450652045)

[6.1 Evaluación de los criterios en la implementación usando Arduino 40](#_Toc450652046)

[6.2 Evaluación de los criterios en la implementación usando Raspberry 41](#_Toc450652047)

[7. Comparación de la implementación de las tecnologías 41](#_Toc450652048)

[8. Conclusiones 44](#_Toc450652049)

# Autores del trabajo, planificación y entrega

## Autores

EL grupo 6 está formado por:

* David Moreno Moreno (Coordinador del grupo)
* Fernando Donaire García
* Sorín Gavrila

Se encargarán de comparar las tecnologías de Arduino y Raspberry.

## Planificación

El peso de este trabajo en la calificación total de la asignatura es de un 30%, por lo tanto, requiere de una dedicación de 45 horas del total de 150 horas de la asignatura.

Al estar formado por 3 integrantes el tiempo total de desarrollo de dicho proyecto será 135 horas, repartidas en el siguiente diagrama de Gantt.

*[TG3 – Arduino&Rapsberry](https://app.ganttpro.com/shared/token/8a64c8d85719ff297de6eeec9b232c698f054c99a9a3a62e7146398537884abc)*

El reparto de tareas ha sido equitativo, diferenciando las tareas del coordinador, por un lado, y la del resto de integrantes del grupo. Cada miembro del grupo se ha especializado en una tecnología para que la información y motivación estuviera orientada al mismo entorno.

## Entrega

Incluimos el enlace (URL) a un repositorio en GitHub donde incorporamos nuestro trabajo y archivos:

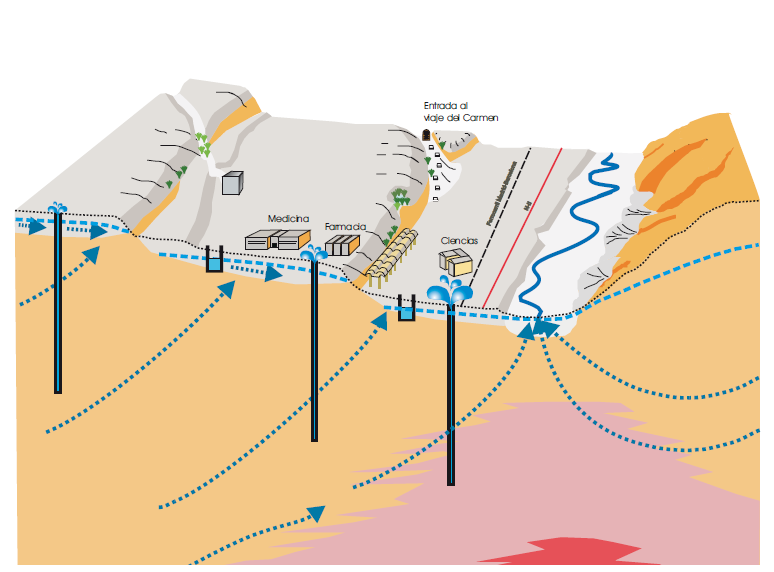
<https://github.com/dmoreno19949/DTE_TG_ARDUINO_RAPSBERRY>

En dicho repositorio hemos incluido una carpeta TG3 donde se encuentran los siguientes archivos:

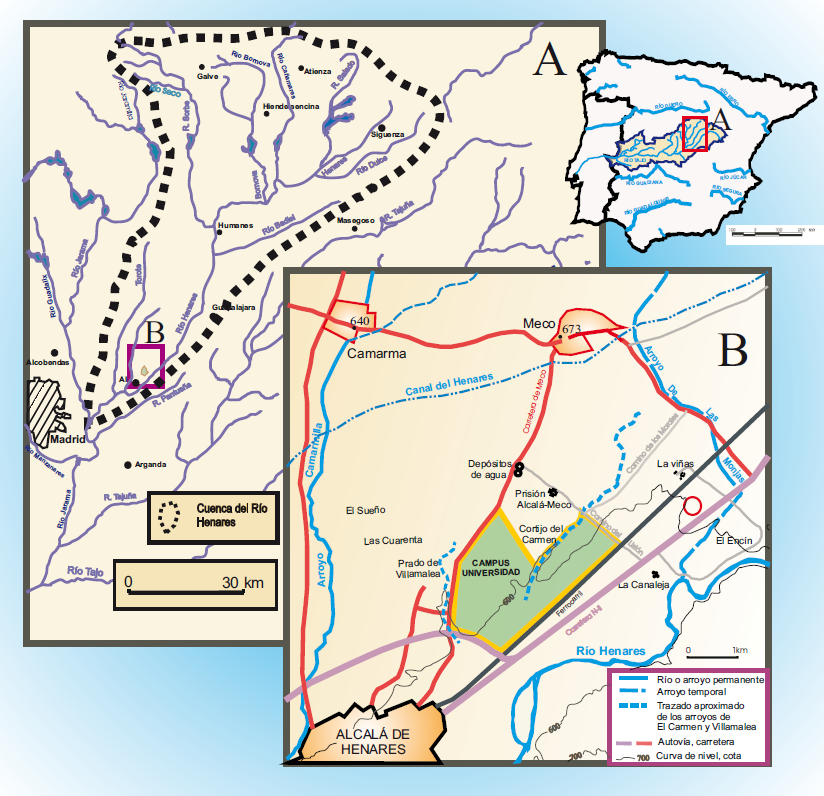
* **Trabajo terminado:** TG3\_final.docx
* **Presentación del trabajo:** TG3\_final.pptx
* **Prototipos obtenidos implementando cada una de las tecnologías** (deben incluir el código fuente y todos los archivos necesarios para la instalación y uso de cada prototipo):
  + PrototipoTecnologiaArduino\_final.zip (o .rar)
  + PrototipoTecnologiaRaspberry\_final.zip (o .rar)

# Requisitos del prototipo a implementar

La Universidad de Alcalá está presentando a concurso público el proyecto **SmartWater.**



“**SmartWater – Fase 1: Desarrollo IoT**” forma parte del plan de medidas de ahorro y sostenibilidad medioambiental “**SaveTheWater-UAH2020**” para calcular el Balance hídrico medio anual aportado por la lluvia y así evitar desgastar los acuíferos en casos de estrés hídrico.



Esta primera fase del proyecto tiene como objetivo implementar una red de sensores a lo largo del campus externo para monitorizar el caudal del agua en los días de lluvia para obtener una media del caudal exacto.

En todos los tejados de todas las facultades se instalarán unas cajas de unas dimensiones y superficies determinadas con una estructura tipo embudo que almacenará el agua de manera temporal hasta que pase por un sensor que mida el caudal del agua.

La estructura y el sensor forman parte de otro proyecto interno de la universidad, pero estos sensores enviarán datos a la plataforma IoT que enviará y almacenará los registros en una base de datos común de la universidad para su posterior tratamiento.



Por tanto, el alcance del proyecto se limita a implementar la parte IoT.

Queda fuera del alcance del proyecto:

* Diseñar e implementar la estructura de recolección del agua
* El sensor lo elige la Universidad
* El procesamiento, estadísticas, etc… forman parte de la Fase 2. Simplemente se implementará una interfaz para ver los datos en tiempo real.

A continuación, se detallan todos los requisitos:

## Requisitos funcionales

Los requisitos funcionales deben ser los mismos para las dos implementaciones.

En la siguiente tabla se indicará el catálogo de requisitos funcionales del sistema.

|  |  |
| --- | --- |
| REQ. | DESCRIPCIÓN |
|  |  |
| Requisitos Funcionales Muestreo de Datos | |
| RFMD01 | Se debe crear una página web, en la que se muestre la información recogida. |
| RFMD02 | Los datos deben ser mostrados gráficamente en un diagrama de líneas. |
| RFMD03 | Los diagramas deben de mostrar los datos recogidos por cada tecnología de forma separada. |
| RFMD04 | El color azul reflejara los datos recogidos por Arduino, mientras que el color rosa reflejara los datos recogidos por Rapsberry. |

## requisitos no funcionales

La siguiente tabla se indicará el catálogo de requisitos no funcionales del sistema.

|  |  |
| --- | --- |
| REQ. | DESCRIPCIÓN |
|  |  |
| Requisitos No Funcionales Placas | |
| RNFP01 | Conectividad Ethernet vía cable Ethernet UTP. |
| RNFP02 | Conexión constante a la corriente. |
|  |  |
| Requisitos No Funcionales Recolección de Datos | |
| RNFRD01 | El lenguaje de programación debe de ser C++ /Phyton. |
| RNFRD02 | Los datos deben ser recolectados en Litros/hora. |
| RNFRD03 | El retardo de la recolección de datos debe ser de 1 segundo. |
| RNFRD04 | Los datos deben ser agrupados en una base de datos estructurada. |
| RNFRD05 | Cada observación contendrá el origen (Arduino o Rapsberry), la medida, un id único y la hora en la que fueron recogidos. |
| RNFRD06 | Los datos han de ser almacenados de forma eficiente y segura. |
|  |  |

# Criterios de comparación en la implementación

En el trabajo TG2 se definieron criterios de comparación de las dos tecnologías a nivel teórico.

En este trabajo se definen criterios para la comparación de la implementación de las tecnologías en la construcción del prototipo de sistema de ejemplo, cuyos requisitos son los establecidos en el apartado 2.

## Criterio 1: Documentación

**Nombre del criterio:** Documentación.

**Descripción:** Documentación encontrada para el desarrollo de la actividad.

**Tipo de valor:** Numérico (unidades).

## Criterio 2: Horas empleadas en el desarrollo del sistema

**Nombre del criterio:** Horas empleadas en el desarrollo del sistema.

**Descripción:** Horas invertidas en la creación del diagrama de clases utilizando el editor de la herramienta.

**Tipo de valor:** Numérico (horas).

## Criterio 3: Líneas de código

**Nombre del criterio:** Líneas de código.

**Descripción:** Número de líneas que componen el código.

**Tipo de valor:** Numérico (líneas).

## Criterio 4: Peso del programa

**Nombre del criterio:** Peso del programa.

**Descripción:** Peso del programa en función de espacio.

**Tipo de valor:** Numérico (Kbyte).

## Criterio 5: Tiempo de arranque de tecnologías

**Nombre del criterio:** Tiempo de arranque de tecnologías.

**Descripción:** Tiempo que pasa desde que se conecta el microcontrolador hasta que envía el primer dato.

**Tipo de valor:** Numérico (segundos).

## Criterio 6: Tiempo verificación configuración Ethernet

**Nombre del criterio:** Tiempo verificación Ethernet.

**Descripción:** Tiempo que pasa desde que se conecta el microcontrolador hasta que es completada la inicialización y verificación de la conectividad a la red.

**Tipo de valor:** Numérico (segundos).

## Criterio 7: Rendimiento en cuanto a funcionamiento interrumpido

**Nombre del criterio:** Rendimiento en cuanto a funcionamiento interrumpido.

**Descripción:** Comportamiento de los microcontroladores de forma interrumpida.

**Tipo de valor:** Numérico (Valoración 0-10).

## Criterio 8: Rendimiento en cuanto a funcionamiento ininterrumpido

**Nombre del criterio:** Rendimiento en cuanto a funcionamiento ininterrumpido.

**Descripción:** Comportamiento de los microcontroladores de forma ininterrumpida.

**Tipo de valor:** Numérico (Valoración 0-10).

## Criterio 9: Recursos necesarios

**Nombre del criterio:** Recursos necesarios.

**Descripción:** Componentes necesitados para la realización de las mismas funciones.

**Tipo de valor:** Numérico (Unidades).

## Criterio 10: Precio

**Nombre del criterio:** Precio.

**Descripción:** Dinero gastado en las tecnologías para realizar las mismas funciones.

**Tipo de valor:** Numérico (Euros).

## Criterio 11: Espacio físico

**Nombre del criterio:** Espacio físico.

**Descripción:** Dimensiones de los microcontroladores.

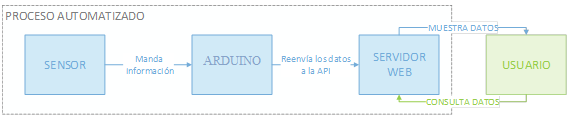
**Tipo de valor:** Numérico (cm).

# Proyecto de implementación de un prototipo del sistema utilizando Arduino

## Documentación de diseño

### Diagrama de uso lógico

Por una parte se detalla el esquema lógico de funcionaiento arduino con los diferentes componentes que se interelaciona.



### Diseño interfaz de usuario

Por otra parte, se muestra de forma gráfica, los diferentes datos recogidos en forma de diagrama de líneas.



## Documentación de construcción

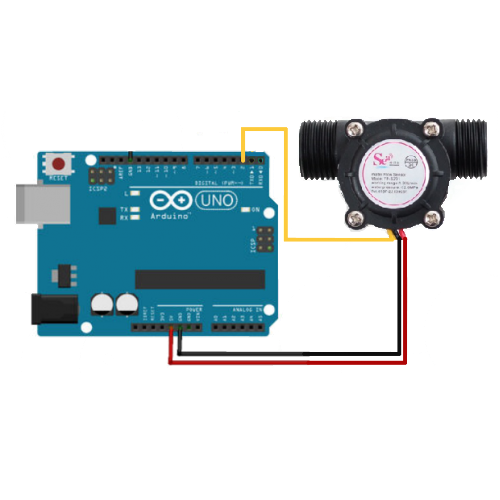
### Diagrama de conexiones Arduino

En primer lugar, hay que recalcar que a diferencia de Rapsberry, Arduino necesita incluir Ethernet Shield si queremos dotar a la placa microcontroladora de conexión a la red.

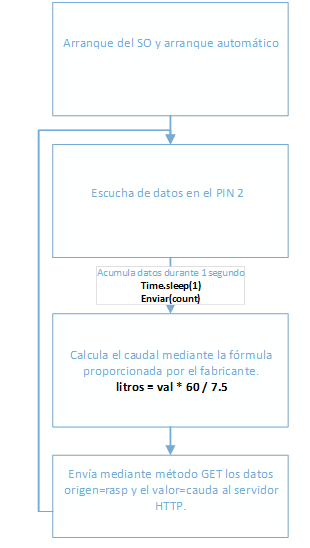
Por ello es preciso la conexión del módulo Ethernet Shield a la placa Arduino, quedando este módulo conectado a través de la clavija rj45 al ordenador, o con una clavija rj45 con conexión a internet a la intranet.



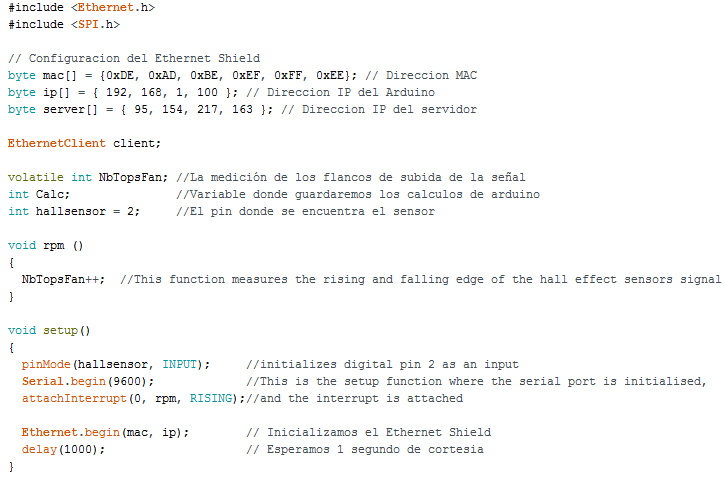
Por otro lado, la conexión del caudalimetro al Arduino se realiza atreves del pin 2 y los pines correspondientes de GND y 5v+. De esta forma el cable rojo y negro dotara el caudalimetro del circuito eléctrico necesario para que funcione. Sin embargo, el cable amarillo será el encargado de transportar los datos con las mediciones pertinentes.



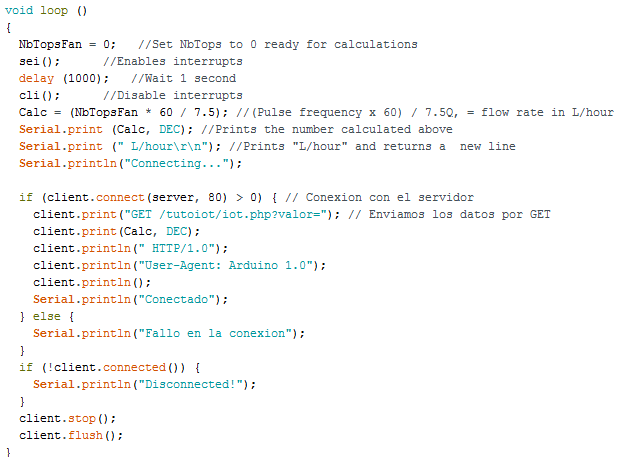
### Diagrama de uso lógico del script de Arduino



Inicialización de variables y configuración de los parámetros de Ethernet.



Bucle ininterrumpido para la recolección de datos y envió a servidor.



### Diagrama de uso lógico de la API

Una vez iniciados los controladores, el servidor lee todas las entradas GET. Cada vez que entra una son recogidas por el archivo iot, el cual ejecuta la siguiente línea de código.



Script completo:

<?php

// iot.php

// Importamos la configuración

require("config.php");

// Leemos los valores que nos llegan por GET

$valor = mysqli\_real\_escape\_string($con, $\_GET['valor']);

$origen = mysqli\_real\_escape\_string($con, $\_GET['origen']);

// Esta es la instrucción para insertar los valores

//$query = "INSERT INTO wp\_valores(valor) VALUES('".$valor."')";

$query = "INSERT INTO wp\_valores(origen, valor) VALUES('".$origen."','".$valor."')";

// Ejecutamos la instrucción

mysqli\_query($con, $query);

mysqli\_close($con);

print "Recibido origen:".$origen." valor:".$valor;

?>

### Diagrama de uso lógico de la API

En la base de datos hemos creado una nueva tabla con las siguientes características:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre tabla: | wp\_valores |  |  |  |
| Campo: | Id | Hora | Origen | valor |
| Tipo: | Int autoincrementado | Timestamp - Fechado de los datos | Varchar | Varchar |
| Valores: | - | fecha | Arduino/Rasp | texto libre |

Script .sql :

-- phpMyAdmin SQL Dump

-- version 4.5.1

-- http://www.phpmyadmin.net

--

-- Servidor: 127.0.0.1

-- Tiempo de generación: 21-04-2016 a las 01:01:47

-- Versión del servidor: 10.1.9-MariaDB

-- Versión de PHP: 5.6.15

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8mb4 \*/;

--

-- Base de datos: `iothings`

--

-- --------------------------------------------------------

--

-- Estructura de tabla para la tabla `wp\_valores`

--

CREATE TABLE `wp\_valores` (

`id` int(11) NOT NULL,

`hora` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

`origen` varchar(5000) NOT NULL,

`valor` varchar(5000) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--

-- Volcado de datos para la tabla `wp\_valores`

--

---OMITIDO----

--

-- Índices para tablas volcadas

--

--

-- Indices de la tabla `wp\_valores`

--

ALTER TABLE `wp\_valores`

ADD PRIMARY KEY (`id`);

-- AUTO\_INCREMENT de las tablas volcadas

--

--

-- AUTO\_INCREMENT de la tabla `wp\_valores`

--

ALTER TABLE `wp\_valores`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3448;

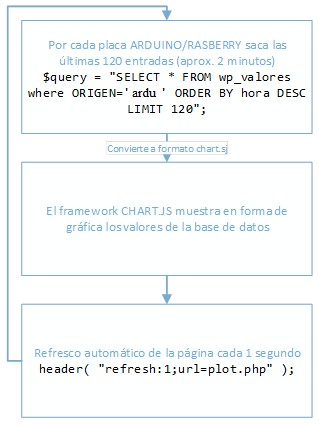
/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

### Diagrama de uso lógico de la interfaz

A través de consultas a la base de datos, organizaremos y graficaremos la información.



Script completo:

<?php

header( "refresh:1;url=plot.php" );

// iot.php

// Importamos la configuración

require("config.php");

// Leemos los valores que nos llegan por GET

// Esta es la instrucción para insertar los valores

//$query = "INSERT INTO wp\_valores(valor) VALUES('".$valor."')";

// Create connection

$conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

$query = "SELECT \* FROM wp\_valores where ORIGEN='arduino' ORDER BY hora DESC LIMIT 120";

$result = $conn->query($query);

$labels = '';

$datas = '';

if ($result->num\_rows > 0) {

// output data of each row

while($row = $result->fetch\_assoc()) {

//echo "id: " . $row["id"]. " - hora: " . $row["hora"]. " - origen: " . $row["origen"]. " - valor: " . $row["valor"]. "<br>";

$labels = $labels.'"'.$row['hora'].'",';

$datas = $datas.'"'.$row['valor'].'",';

}

} else {

echo "0 results";

}

$query = "SELECT \* FROM wp\_valores where ORIGEN='rasp' ORDER BY hora DESC LIMIT 120";

$result = $conn->query($query);

$labelsR = '';

$datasR = '';

if ($result->num\_rows > 0) {

// output data of each row

while($row = $result->fetch\_assoc()) {

//echo "id: " . $row["id"]. " - hora: " . $row["hora"]. " - origen: " . $row["origen"]. " - valor: " . $row["valor"]. "<br>";

$labelsR = $labelsR.'"'.$row['hora'].'",';

$datasR = $datasR.'"'.$row['valor'].'",';

}

} else {

echo "0 results";

}

$conn->close();

?>

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Lumino - Charts</title>

<link href="css/bootstrap.min.css" rel="stylesheet">

<link href="css/datepicker3.css" rel="stylesheet">

<link href="css/styles.css" rel="stylesheet">

<!--Icons-->

<script src="js/lumino.glyphs.js"></script>

<!--[if lt IE 9]>

<script src="js/html5shiv.js"></script>

<script src="js/respond.min.js"></script>

<![endif]-->

</head>

<body style="font-size:6px;">

<div class="col-sm-12">

<div class="row">

<div class="col-lg-12">

<div class="panel panel-default">

<div class="panel-heading">ARDUINO</div>

<div class="panel-body">

<div class="canvas-wrapper">

<canvas class="main-chart" id="line-chart" height="200" width="900"></canvas>

</div>

</div>

</div>

</div>

</div><!--/.row-->

<div class="row">

<div class="col-lg-12">

<div class="panel panel-default">

<div class="panel-heading">RASPBERRY</div>

<div class="panel-body">

<div class="canvas-wrapper">

<canvas class="main-chart" id="line-chart1" height="200" width="900"></canvas>

</div>

</div>

</div>

</div>

</div><!--/.row-->

</div> <!--/.main-->

<script src="js/jquery-1.11.1.min.js"></script>

<script src="js/bootstrap.min.js"></script>

<script src="js/chart.min.js"></script>

<script src="js/chart-data.js"></script>

<script src="js/easypiechart.js"></script>

<script src="js/easypiechart-data.js"></script>

<script src="js/bootstrap-datepicker.js"></script>

<script>

</script>

<script>

var randomScalingFactor = function(){ return Math.round(Math.random()\*1000)};

var lineChartData = {

labels : [<?php print $labels;?>],

datasets : [

{

label: "Arduino",

fillColor : "rgba(0,220,220,0.2)",

strokeColor : "rgba(0,220,220,1)",

pointColor : "rgba(0,220,220,1)",

pointStrokeColor : "#fff",

pointHighlightFill : "#fff",

pointHighlightStroke : "rgba(0,220,220,1)",

data : [<?php print $datas;?>]

}

]

}

var lineChartData1 = {

labels : [<?php print $labelsR;?>],

datasets : [

{

label: "RASPBERRY",

fillColor : "rgba(220,0,220,0.2)",

strokeColor : "rgba(220,0,220,1)",

pointColor : "rgba(220,0,220,1)",

pointStrokeColor : "#fff",

pointHighlightFill : "#fff",

pointHighlightStroke : "rgba(220,0,220,1)",

data : [<?php print $datasR;?>]

}

]

}

window.onload = function(){

var chart1 = document.getElementById("line-chart").getContext("2d");

window.myLine = new Chart(chart1).Line(lineChartData, {

responsive: true,

animation: false

});

var chart2 = document.getElementById("line-chart1").getContext("2d");

window.myLine2 = new Chart(chart2).Line(lineChartData1, {

responsive: true,

animation: false

});

};

!function ($) {

$(document).on("click","ul.nav li.parent > a > span.icon", function(){

$(this).find('em:first').toggleClass("glyphicon-minus");

});

$(".sidebar span.icon").find('em:first').addClass("glyphicon-plus");

}(window.jQuery);

$(window).on('resize', function () {

if ($(window).width() > 768) $('#sidebar-collapse').collapse('show')

})

$(window).on('resize', function () {

if ($(window).width() <= 767) $('#sidebar-collapse').collapse('hide')

})

</script>

</body>

</html>

## Documentación de pruebas

Casos de prueba establecidos y resultados de las pruebas y acciones de corrección.

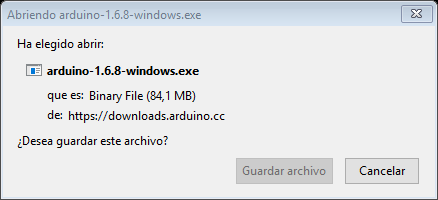
|  |  |  |  |
| --- | --- | --- | --- |
| Prueba y pasos | Objetivo | Pasos | Estado |
| Conectividad Ethernet | Ver que tenemos conexión con el servidor | Configurar parámetros tanto en servidor y como Arduino. Conectar cables. Hacer pings. | Correcto. |
| Prueba del programa | Ver que podemos obtener datos | Conectar el sensor a los pines correspondientes.  Habilitar salida para depuración por consola.  Lanzar script. Mover el sensor para generar datos. Los datos aparecen por consola. | Correcto. |
| Prueba de conexión con el servidor HTTP | Ver que se reciben datos | Conectar el sensor a los pines correspondientes.  Habilitar salida para depuración por consola.  Habilitar código para enviar peticiones GET. Lanzar script. Mover el sensor para generar datos. Los datos aparecen por consola. Los datos aparecen en la BBDD. | Problemas iniciales con los parámetros desde Arduino, corregido.  Correcto. |
| Probar que el programa arranca solo | Ver que el script se ejecuta de forma autónoma después del reinicio. | Pulsar el botón de reinicio. Ver que se reciben datos. | Correcto. |
| Probar que la conexión se recupera tras una desconexión del cable Ethernet | Ver que se reciben datos | Con el script lanzado de forma automática quitar el cable. La base de datos ya no se actualiza. Reconectar el cable. La base de datos ya se vuelve a actualizar con normalidad sin intervención del usuario. | El script nunca se detiene, aunque pierda la conexión Ethernet.  Correcto. |

## Documentación de instalación

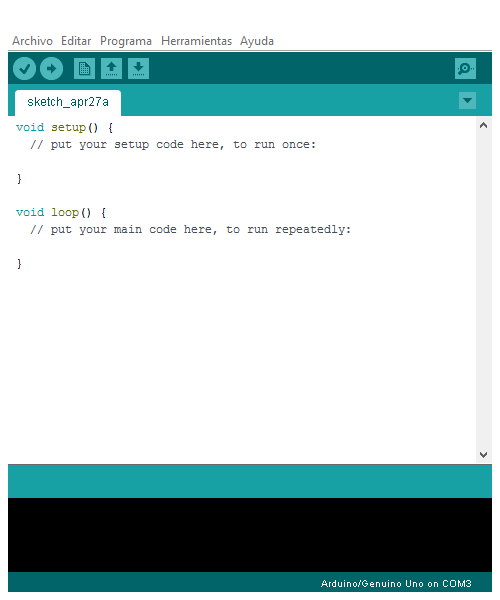
Arduino a diferencia de Rapsberry, no necesita ninguna imagen ISO integrada en la placa microcontroladora.

Lo que si te proporciona Arduino es un software para la programación, revisión, envió de comandos a través terminales, elección de puertos muy recomendable para cualquier proyecto que se desee realizar.

Para ello entraremos en <https://www.arduino.cc/en/Main/Software> , elegiremos el software en función de nuestro sistema operativo, y seguiremos las pautas que nos indican.



Finalmente habremos instalado el programa con el siguiente diseño:



## Manual de usuario

1. Conectar cables y sensor.
2. Arrancar servidor Xampp.



3) Acceder a la dirección <http://localhost/graficos/plot.php>

Las gráficas se refrescarán automáticamente según se van recibiendo los datos.



# Proyecto de implementación de un prototipo del sistema utilizando Raspberry

Se trata de incluir en este apartado la documentación del desarrollo del proyecto de implementación, utilizando la tecnología B, del sistema cuyos requisitos funcionales se enumeraron en el apartado 2.

## Documentación de diseño

Hay que incluir la descripción del diseño del prototipo, incluyendo diagramas, y el diseño de la interfaz de usuario.

### Diagrama de uso lógico

Se detalla las funciones y acciones que tiene cada componente.



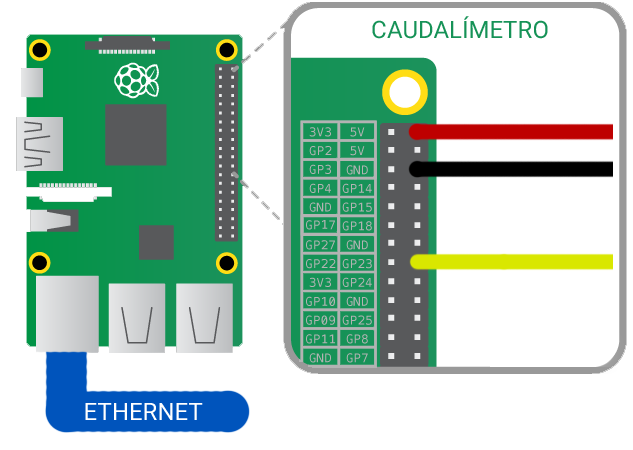
### Diseño interfaz de usuario



## Documentación de construcción

Hay que incluir una descripción de la construcción del prototipo, incluyendo algún extracto de código fuente. No es necesario todo el código. Sólo algún extracto para ver cómo se ha comentado.

### Diagrama de conexiones Raspberry



### Diagrama de uso lógico del script de Raspberry



Script completo:

pi@raspberrypi ~ $ cat sensorGET.py

#!/usr/bin/env python

import urllib

import urllib2

def enviar(val):

litros = val \* 60 / 7.5

values = {'origen' : 'rasp' ,

'valor' : litros }

data = urllib.urlencode(values)

url = 'http://192.168.10.1/iot.php?' + data

response = urllib2.urlopen(url).read()

print data

print response

print litros

print ' \t por minuto'

import RPi.GPIO as GPIO

import time, sys

FLOW\_SENSOR = 23

GPIO.setmode(GPIO.BCM)

GPIO.setup(FLOW\_SENSOR, GPIO.IN, pull\_up\_down = GPIO.PUD\_UP)

global count

count = 0

def countPulse(channel):

global count

count = count+1

GPIO.add\_event\_detect(FLOW\_SENSOR, GPIO.FALLING, callback=countPulse)

while True:

try:

time.sleep(1)

enviar(count)

count = 0

except KeyboardInterrupt:

print '\ncaught keyboard interrupt!, bye'

GPIO.cleanup()

sys.exit()

### Diagrama de uso lógico de la API



Script completo:

<?php

// iot.php

// Importamos la configuración

require("config.php");

// Leemos los valores que nos llegan por GET

$valor = mysqli\_real\_escape\_string($con, $\_GET['valor']);

$origen = mysqli\_real\_escape\_string($con, $\_GET['origen']);

// Esta es la instrucción para insertar los valores

//$query = "INSERT INTO wp\_valores(valor) VALUES('".$valor."')";

$query = "INSERT INTO wp\_valores(origen, valor) VALUES('".$origen."','".$valor."')";

// Ejecutamos la instrucción

mysqli\_query($con, $query);

mysqli\_close($con);

print "Recibido origen:".$origen." valor:".$valor;?>

### Diagrama de la base de datos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre tabla: | wp\_valores |  |  |  |
| Campo: | Id | Hora | Origen | valor |
| Tipo: | Int autoincrementado | Timestamp - Fechado de los datos | Varchar | Varchar |
| Valores: | - | fecha | Arduino/Rasp | texto libre |

Script .sql :

-- phpMyAdmin SQL Dump

-- version 4.5.1

-- http://www.phpmyadmin.net

--

-- Servidor: 127.0.0.1

-- Tiempo de generación: 21-04-2016 a las 01:01:47

-- Versión del servidor: 10.1.9-MariaDB

-- Versión de PHP: 5.6.15

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8mb4 \*/;

--

-- Base de datos: `iothings`

--

-- --------------------------------------------------------

--

-- Estructura de tabla para la tabla `wp\_valores`

--

CREATE TABLE `wp\_valores` (

`id` int(11) NOT NULL,

`hora` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

`origen` varchar(5000) NOT NULL,

`valor` varchar(5000) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--

-- Volcado de datos para la tabla `wp\_valores`

--

---OMITIDO----

--

-- Índices para tablas volcadas

--

--

-- Indices de la tabla `wp\_valores`

--

ALTER TABLE `wp\_valores`

ADD PRIMARY KEY (`id`);

--

-- AUTO\_INCREMENT de las tablas volcadas

--

--

-- AUTO\_INCREMENT de la tabla `wp\_valores`

--

ALTER TABLE `wp\_valores`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3448;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

### Diagrama de uso lógico de la Interfaz de Usuario



Script completo:

<?php

header( "refresh:1;url=plot.php" );

// iot.php

// Importamos la configuración

require("config.php");

// Leemos los valores que nos llegan por GET

// Esta es la instrucción para insertar los valores

//$query = "INSERT INTO wp\_valores(valor) VALUES('".$valor."')";

// Create connection

$conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

$query = "SELECT \* FROM wp\_valores where ORIGEN='arduino' ORDER BY hora DESC LIMIT 120";

$result = $conn->query($query);

$labels = '';

$datas = '';

if ($result->num\_rows > 0) {

// output data of each row

while($row = $result->fetch\_assoc()) {

//echo "id: " . $row["id"]. " - hora: " . $row["hora"]. " - origen: " . $row["origen"]. " - valor: " . $row["valor"]. "<br>";

$labels = $labels.'"'.$row['hora'].'",';

$datas = $datas.'"'.$row['valor'].'",';

}

} else {

echo "0 results";

}

$query = "SELECT \* FROM wp\_valores where ORIGEN='rasp' ORDER BY hora DESC LIMIT 120";

$result = $conn->query($query);

$labelsR = '';

$datasR = '';

if ($result->num\_rows > 0) {

// output data of each row

while($row = $result->fetch\_assoc()) {

//echo "id: " . $row["id"]. " - hora: " . $row["hora"]. " - origen: " . $row["origen"]. " - valor: " . $row["valor"]. "<br>";

$labelsR = $labelsR.'"'.$row['hora'].'",';

$datasR = $datasR.'"'.$row['valor'].'",';

}

} else {

echo "0 results";

}

$conn->close();

?>

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Lumino - Charts</title>

<link href="css/bootstrap.min.css" rel="stylesheet">

<link href="css/datepicker3.css" rel="stylesheet">

<link href="css/styles.css" rel="stylesheet">

<!--Icons-->

<script src="js/lumino.glyphs.js"></script>

<!--[if lt IE 9]>

<script src="js/html5shiv.js"></script>

<script src="js/respond.min.js"></script>

<![endif]-->

</head>

<body style="font-size:6px;">

<div class="col-sm-12">

<div class="row">

<div class="col-lg-12">

<div class="panel panel-default">

<div class="panel-heading">ARDUINO</div>

<div class="panel-body">

<div class="canvas-wrapper">

<canvas class="main-chart" id="line-chart" height="200" width="900"></canvas>

</div>

</div>

</div>

</div>

</div><!--/.row-->

<div class="row">

<div class="col-lg-12">

<div class="panel panel-default">

<div class="panel-heading">RASPBERRY</div>

<div class="panel-body">

<div class="canvas-wrapper">

<canvas class="main-chart" id="line-chart1" height="200" width="900"></canvas>

</div>

</div>

</div>

</div>

</div><!--/.row-->

</div> <!--/.main-->

<script src="js/jquery-1.11.1.min.js"></script>

<script src="js/bootstrap.min.js"></script>

<script src="js/chart.min.js"></script>

<script src="js/chart-data.js"></script>

<script src="js/easypiechart.js"></script>

<script src="js/easypiechart-data.js"></script>

<script src="js/bootstrap-datepicker.js"></script>

<script>

</script>

<script>

var randomScalingFactor = function(){ return Math.round(Math.random()\*1000)};

var lineChartData = {

labels : [<?php print $labels;?>],

datasets : [

{

label: "Arduino",

fillColor : "rgba(0,220,220,0.2)",

strokeColor : "rgba(0,220,220,1)",

pointColor : "rgba(0,220,220,1)",

pointStrokeColor : "#fff",

pointHighlightFill : "#fff",

pointHighlightStroke : "rgba(0,220,220,1)",

data : [<?php print $datas;?>]

}

]

}

var lineChartData1 = {

labels : [<?php print $labelsR;?>],

datasets : [

{

label: "RASPBERRY",

fillColor : "rgba(220,0,220,0.2)",

strokeColor : "rgba(220,0,220,1)",

pointColor : "rgba(220,0,220,1)",

pointStrokeColor : "#fff",

pointHighlightFill : "#fff",

pointHighlightStroke : "rgba(220,0,220,1)",

data : [<?php print $datasR;?>]

}

]

}

window.onload = function(){

var chart1 = document.getElementById("line-chart").getContext("2d");

window.myLine = new Chart(chart1).Line(lineChartData, {

responsive: true,

animation: false

});

var chart2 = document.getElementById("line-chart1").getContext("2d");

window.myLine2 = new Chart(chart2).Line(lineChartData1, {

responsive: true,

animation: false

});

};

!function ($) {

$(document).on("click","ul.nav li.parent > a > span.icon", function(){

$(this).find('em:first').toggleClass("glyphicon-minus");

});

$(".sidebar span.icon").find('em:first').addClass("glyphicon-plus");

}(window.jQuery);

$(window).on('resize', function () {

if ($(window).width() > 768) $('#sidebar-collapse').collapse('show')

})

$(window).on('resize', function () {

if ($(window).width() <= 767) $('#sidebar-collapse').collapse('hide')

})

</script>

</body>

</html>

## Documentación de pruebas

Casos de prueba establecidos y resultados de las pruebas y acciones de corrección.

|  |  |  |  |
| --- | --- | --- | --- |
| Prueba y pasos | Objetivo | Pasos | Estado |
| Arranque del sistema operativo | Ver que todo arranca bien | Conectar todos los cables y SO en tarjeta de memoria. | Correcto |
| Conectividad Ethernet | Ver que tenemos conexión con el servidor | Configurar Ios en servidor y Raspberry. Conectar cables. Hacer pings. Hacer ssh desde servidor a Raspberry | Correcto |
| Prueba del programa | Ver que podemos obtener datos | Conectar el sensor a los pines correspondientes.  Habilitar salida para depuración por consola.  Lanzar script. Mover el sensor para generar datos. Los datos aparecen por consola. | Correcto |
| Prueba de conexión con el servidor HTTP | Ver que se reciben datos | Conectar el sensor a los pines correspondientes.  Habilitar salida para depuración por consola.  Habilitar código para enviar peticiones GET. Lanzar script. Mover el sensor para generar datos. Los datos aparecen por consola. Los datos aparecen en la BBDD. | Problemas iniciales con los parámetros desde Raspberry, corregido.  Correcto |
| Probar que el programa arranca solo | Ver que el script se ejecuta de forma autónoma después del reinicio. | Añadir la línea correspondiente para el arranque.  Reiniciar. Ver que se reciben datos. | Hemos probado varias soluciones de arranque, la primera dependía de si el usuario hacía login (perfil) y la segunda iba con el arranque del SO (hemos elegido esta). Correcto |
| Probar que la conexión se recupera tras una desconexión del cable ethernet | Ver que se reciben datos | Con el script lanzado de forma automática quitar el cable. La base de datos ya no se actualiza. Reconectar el cable. La base de datos ya se vuelve a actualizar con normalidad sin intervención del usuario. | Correcto, el script nunca se detiene aunque pierda la conexión Ethernet. |

## Documentación de instalación

Descripción suficiente para que una persona que no ha participado en el proyecto pueda instalar el prototipo.

Requisitos: Tarjeta microSD de 16gb.

### Instalar el sistema operativo de Raspberry

1) Descargar imagen del SO desde <https://www.raspberrypi.org/downloads/raspbian/>

RASPBIAN JESSIE LITE

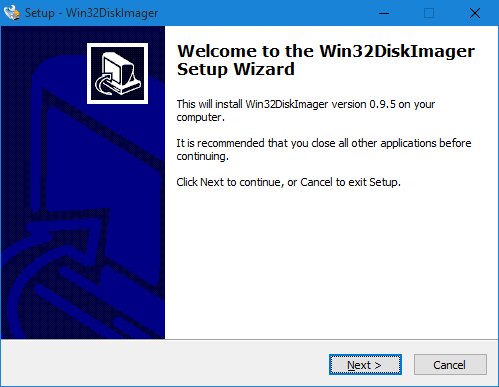
Minimal image based on Debian Jessie

Version:March 2016

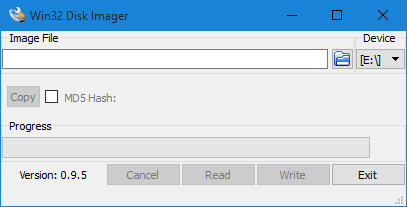
Release date:2016-03-18

Kernel version:4.1

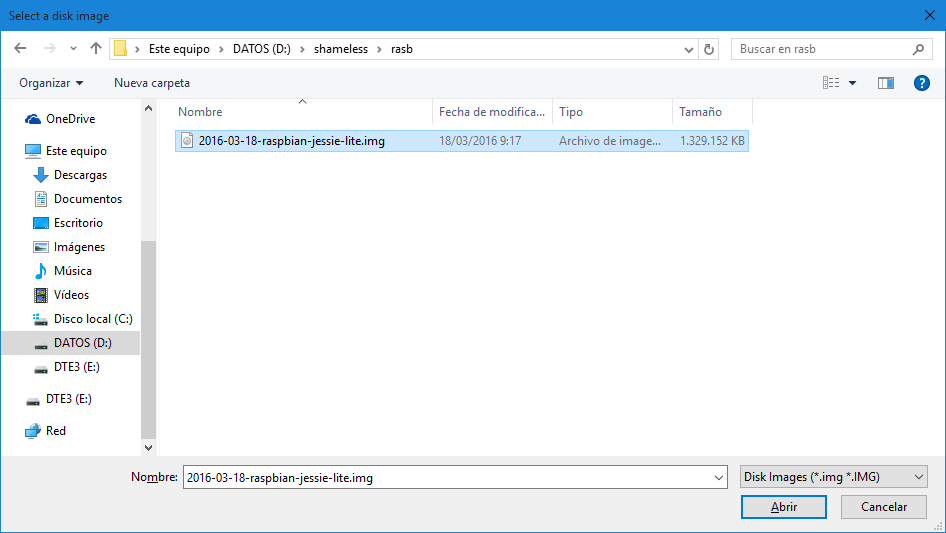
2) Instalar Win32DiskImager.



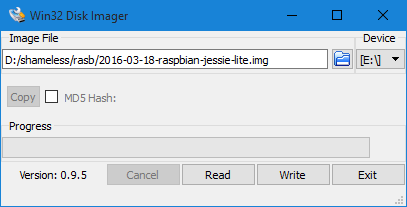
3) Ejecutar.



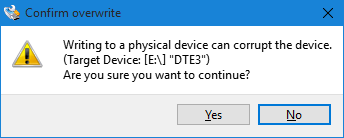
4) Seleccionar la imagen de Raspberian.



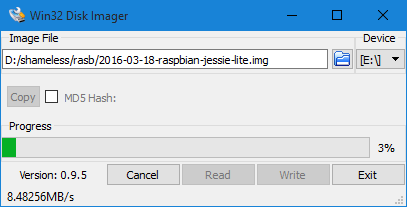
5) Seleccionar la opción WRITE para escribir la imagen en la tarjeta SD.

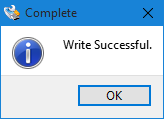


6) Confirmar borrado de la tarjeta.



7) Esperar a que el proceso acabe.





### Configurar el sistema operativo de Raspberry

Insertamos la tarjeta de memoria, teclado, ratón, cable hdmi y la alimentación. La placa arrancará automáticamente.

Hacemos login en el SO. Por defecto el usuario es **pi** y la contraseña **raspberry**.

#### Configurar la interfaz de red

Usando el editor vi configuramos la interfaz de la siguiente manera y guardamos los cambios mediante el comando **:wq!**.

pi@raspberrypi ~ $ sudo vi /etc/network/interfaces

auto lo

iface lo inet loopback

#iface eth0 inet dhcp

iface eth0 inet static

address 192.168.10.10

netmask 255.255.255.0

gateway 192.168.10.1

broadcast 255.255.255.255

allow-hotplug wlan0

iface wlan0 inet manual

wpa-roam /etc/wpa\_supplicant/wpa\_supplicant.conf

iface default inet dhcp

Reiniciamos para aplicar los cambios.

pi@raspberrypi ~ $ sudo reboot

Conectamos el cable de red y comprobamos que tenemos conectividad al servidor HTTP 192.168.10.1.

pi@raspberrypi ~ $ ping 192.168.10.1

#### Copiar el script del sensor

Tenemos 2 opciones, copiar el fichero sensorGET.py a la tarjeta y de ahí copiarlo al directorio del usuario (/home/pi) o aprovechar la conectividad de red y conectarnos por consola y crear el fichero.

pi@raspberrypi ~ $ cd /home/pi

pi@raspberrypi ~ $ vi sensorGET.py

#!/usr/bin/env python

import urllib

import urllib2

def enviar(val):

litros = val \* 60 / 7.5

values = {'origen' : 'rasp' ,

'valor' : litros }

data = urllib.urlencode(values)

url = 'http://192.168.10.1/iot.php?' + data

response = urllib2.urlopen(url).read()

print data

print response

print litros

print ' \t por minuto'

import RPi.GPIO as GPIO

import time, sys

FLOW\_SENSOR = 23

GPIO.setmode(GPIO.BCM)

GPIO.setup(FLOW\_SENSOR, GPIO.IN, pull\_up\_down = GPIO.PUD\_UP)

global count

count = 0

def countPulse(channel):

global count

count = count+1

GPIO.add\_event\_detect(FLOW\_SENSOR, GPIO.FALLING, callback=countPulse)

while True:

try:

time.sleep(1)

enviar(count)

count = 0

except KeyboardInterrupt:

print '\ncaught keyboard interrupt!, bye'

GPIO.cleanup()

sys.exit()

Asignarle permisos de ejecución.

pi@raspberrypi ~ $ chmod 777 sensorGET.py

#### Hacer que el script del sensor arranque automáticamente con el SO

Desde la consola editamos el fichero /etc/rc.local.

pi@raspberrypi ~ $ cat /etc/rc.local

#!/bin/sh -e

#

# rc.local

#

# This script is executed at the end of each multiuser runlevel.

# Make sure that the script will "exit 0" on success or any other

# value on error.

#

# In order to enable or disable this script just change the execution

# bits.

#

# By default this script does nothing.

# Print the IP address

\_IP=$(hostname -I) || true

if [ "$\_IP" ]; then

printf "My IP address is %s\n" "$\_IP"

fi

printf "##ARRANCAMOS SENSOR## "

sudo python /home/pi/sensorGET.py&

printf "##ARRANCAMOS SENSOR## "

exit 0

pi@raspberrypi ~ $

Reiniciamos para aplicar los cambios.

pi@raspberrypi ~ $ sudo reboot

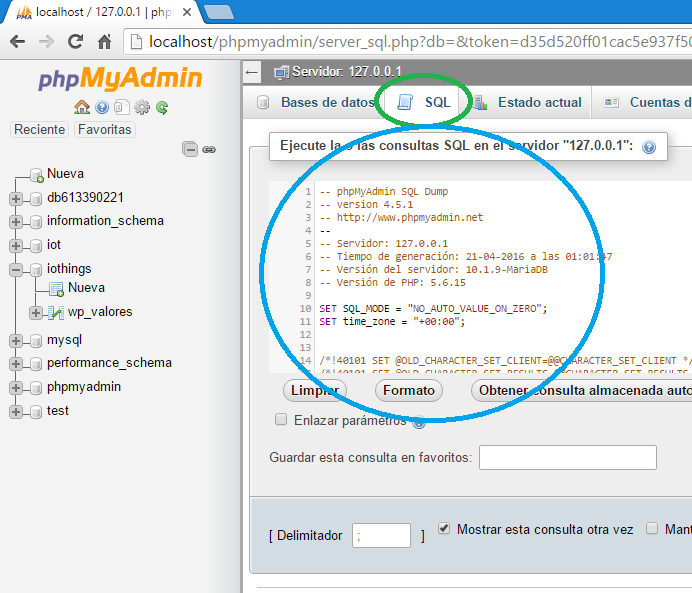
### Configurar servidor web

1) Arrancamos Apache y MySQL desde el panel de control de Xampp.



2) Seleccionamos Mysql Admin para que nos lleve a la página de gestión de la base de datos.

3) Seleccionamos la pestaña SQL e importamos el backup de la base de datos.



4) Configuramos los parámetros del servidor en el fichero C:/xampp/htdocs/config.php

<?php

// config.php

// Credenciales

$dbhost = "localhost";

$dbuser = "iothings";

$dbpass = "iothings";

$dbname = "iothings";

// Conexión con la base de datos

$con = mysqli\_connect($dbhost, $dbuser, $dbpass, $dbname);

?>

## Manual de usuario

Descripción suficiente para que una persona que no ha participado en el proyecto pueda utilizar toda la funcionalidad que ofrece el prototipo. Que debe coincidir con los requisitos funcionales incluidos en el apartado 2.

1. Conectar cables y sensor.
2. Arrancar servidor Xampp.



3) Acceder a la dirección <http://localhost/graficos/plot.php>

Las gráficas se refrescarán automáticamente según se van recibiendo los datos.



# Comparación de las dos implementaciones

Se trata de dar valores a los criterios de comparación definidos en el apartado 3 sobre la implementación de cada uno de los prototipos.

## Evaluación de los criterios en la implementación usando Arduino

Se incluye una tabla donde se puede observar los criterios de evaluación que se han utilizado, su evaluación y posibles comentarios para su mejor entendimiento.

|  |  |
| --- | --- |
| CRITERIO | EVALUACIÓN |
| Criterio 1: Documentación | 5 páginas web encontradas para poder desarrollar el proyecto |
| Criterio 2: Horas empleadas en el desarrollo del sistema | 20 Horas + 5 Horas (pruebas, estabilidad, etc.) = 25 Horas |
| Criterio 3: Líneas de código | 62 líneas (código Arduino) + 203 líneas (código plataforma web) = 265 líneas |
| Criterio 4: Peso del programa | 2 Kb (Arduino) + 6 Kb (plataforma web) = 8 Kb |
| Criterio 5: Tiempo de arranque de tecnologías | 5 segundos |
| Criterio 6: Tiempo verificación configuración Ethernet | 5 segundos |
| Criterio 7: Rendimiento en cuanto a funcionamiento interrumpido | 10 (Se resetea con un botón) |
| Criterio 8: Rendimiento en cuanto a funcionamiento ininterrumpido | 10 (No se han detectado problemas de conexión) |
| Criterio 9: Recursos necesarios | Placa Arduino, Ethernet Shield, Caudalimetro, Ordenador, Servidor web |
| Criterio 10: Precio | 26€ (Arduino) + 9€ (Ethernet Shield) + 8€ (Caudalimetro) = 43€ |
| Criterio 11: Espacio físico | 68,58mmx53,34mm (Arduino) + 70mmx52mmx25mm (Ethernet Shield) + 62mmx35,99mmx 36mm (Caudalimetro) |

## Evaluación de los criterios en la implementación usando Raspberry

Se incluye una tabla donde se puede observar los criterios de evaluación que se han utilizado, su evaluación y posibles comentarios para su mejor entendimiento.

|  |  |
| --- | --- |
| CRITERIO | EVALUACIÓN |
| Criterio 1: Documentación | 3 páginas web encontradas para poder desarrollar el proyecto |
| Criterio 2: Horas empleadas en el desarrollo del sistema | 30 Horas + 5 Horas (pruebas, estabilidad, etc.) = 35 Horas |
| Criterio 3: Líneas de código | 45 líneas (código Raspberry) + 203 líneas (código plataforma web) = 248 líneas |
| Criterio 4: Peso del programa | 2 Kb (Raspberry) + 6 Kb (plataforma web) = 8 Kb |
| Criterio 5: Tiempo de arranque de tecnologías | 30 segundos |
| Criterio 6: Tiempo verificación configuración Ethernet | 5 segundos |
| Criterio 7: Rendimiento en cuanto a funcionamiento interrumpido | 5 (Se resetea conectándose por consola para reiniciar la interfaz o el sistema operativo) |
| Criterio 8: Rendimiento en cuanto a funcionamiento ininterrumpido | 10 (No se han detectado problemas de conexión) |
| Criterio 9: Recursos necesarios | Placa Raspberry, Caudalimetro, Ordenador, Servidor web |
| Criterio 10: Precio | 35€ (Raspberry) + 8€ (Caudalimetro) = 43€ |
| Criterio 11: Espacio físico | 85,60mmx56mm (Raspberry) + 62mmx35,99mmx 36mm (Caudalímetro) |

# Comparación de la implementación de las tecnologías

Se incluye una tabla resumen donde cruzamos los criterios y los valores de cada tecnología estudiada. También se añade a dicha tabla una columna de comentarios sobre la comparación.

|  |  |  |  |
| --- | --- | --- | --- |
| CRITERIOS | ARDUINO | RASPBERRY | COMENTARIOS |
| Criterio 1: Documentación | 5 páginas web encontradas para poder desarrollar el proyecto | 3 páginas web encontradas para poder desarrollar el proyecto | Hay más documentación de proyectos hardware para Arduino que para Raspberry, por lo que es más sencillo encontrar dicha documentación. |
| Criterio 2: Horas empleadas en el desarrollo del sistema | 20 Horas + 5 Horas (pruebas, estabilidad, etc.) = 25 Horas | 30 Horas + 5 Horas (pruebas, estabilidad, etc.) = 35 Horas | Se comprueba que el número de horas es menor para realizar el proyecto en Arduino, debido a la mayor cantidad de documentación. |
| Criterio 3: Líneas de código | 62 líneas (código Arduino) + 203 líneas (código plataforma web) = 265 líneas | 45 líneas (código Raspberry) + 203 líneas (código plataforma web) = 248 líneas | El número de líneas del programa de Raspberry es menor, pero no es una diferencia significativa. |
| Criterio 4: Peso del programa | 2 Kb (Arduino) + 6 Kb (plataforma web) = 8 Kb | 2 Kb (Raspberry) + 6 Kb (plataforma web) = 8 Kb | El peso de los programas es el mismo. |
| Criterio 5: Tiempo de arranque de tecnologías | 5 segundos | 30 segundos | Arduino es más rápido debido a que trabaja directamente en el hardware, Raspberry, por su parte, tiene que inicializar el software. |
| Criterio 6: Tiempo verificación configuración Ethernet | 5 segundos | 5 segundos | La conexión a Internet y la transmisión de datos es igual en ambos casos. |
| Criterio 7: Rendimiento en cuanto a funcionamiento interrumpido | 10 (Se resetea con un botón) | 5 (Se resetea conectándose por consola para reiniciar la interfaz o el sistema operativo) | Arduino tiene más puntuación que Raspberry debido a que la forma de resetearlo es mucho más sencilla y rápida, por lo que el tiempo utilizado es mucho menor. |
| Criterio 8: Rendimiento en cuanto a funcionamiento ininterrumpido | 10 (No se han detectado problemas de conexión) | 10 (No se han detectado problemas de conexión) | Las dos tienen la máxima puntuación debido a que no se detectan problemas de conexión. |
| Criterio 9: Recursos necesarios | Placa Arduino, Ethernet Shield, Caudalimetro, Ordenador, Servidor web | Placa Raspberry, Caudalimetro, Ordenador, Servidor web | Arduino necesita una placa de Ethernet complementaria. |
| Criterio 10: Precio | 26€ (Arduino) + 9€ (Ethernet Shield) + 8€ (Caudalimetro) = 43€ | 35€ (Raspberry) + 8€ (Caudalimetro) = 43€ | El precio es el mismo debido que aunque Raspberry es más cara que Arduino, y el precio del caudalimetro es el mismo, Arduino necesita una placa complementaria de Ethernet. |
| Criterio 11: Espacio físico | 68,58mmx53,34mm (Arduino) + 70mmx52mmx25mm (Ethernet Shield) + 62mmx35,99mmx 36mm (Caudalimetro) | 85,60mmx56mm (Raspberry) + 62mmx35,99mmx 36mm (Caudalimetro) | Ocupa más espacio físico el proyecto creado en Arduino debido a la instalación de la placa de Ethernet. |

# Conclusiones

A partir de la información obtenida en la comparación se pueden sacar diversas conclusiones, que se describirán a continuación.

La principal de todas, es que el proyecto presentado es la elaboración de un caudalimetro, por lo que es un proyecto más relacionado con el hardware, entonces la mejor tecnología para elaborarlo es la tecnología Arduino.

A esta idea final se llega debido a que hay más documentación para programar y elaborar proyectos hardware de Arduino, y aunque el proyecto en sí no estaba desarrollado, se ha podido desarrollar más fácilmente. Toda esta mayor documentación ha provocado que el tiempo empleado para el desarrollo del proyecto en Arduino sea inferior que hacerlo en Raspberry. También, el tiempo de arranque y funcionamiento de Arduino es mucho menor que el de Raspberry, debido a que esta debe inicializarse desde el software y no directamente desde el hardware como Arduino. Otro punto a favor de Arduino importante para el desarrollo del proyecto, es el rendimiento en cuanto a funcionamiento interrumpido, ya que resetear Arduino es mucho más sencillo y rápido porque posee un botón, no como Raspberry que se resetea conectándose por consola para reiniciar la interfaz o el sistema operativo.

En otros criterios elegidos para la comparación, el resultado de los mismos ha sido igual, como son el peso del programa, la conexión a Internet y transmisión de datos, el rendimiento en cuanto a funcionamiento ininterrumpido, ya que no se detectan problemas de conexión, y el precio, dado que, aunque Arduino es más barato, y el caudalimetro es el mismo, se necesita una placa de Ethernet para poder la conexión con el servidor web.

En los demás criterios comparados, la placa Raspberry partiría como mejor opción, pero la diferencia no es significativa ya que en el número de líneas de los códigos de los programas la diferencia es pequeña, el espacio físico de los proyectos es diferente pero no tiene relevancia en el proyecto y para poder realizar el proyecto, Raspberry necesita un componente menos (Arduino necesita una placa Ethernet complementaria), pero su utilización no afecta en demasía ya que sólo afectaría al tamaño puesto que lo que ocurre con los precios es que se igualan.